Network Node Pilot Project

Beta Phase:

Report on Project Results and Next Steps

PRE-FINAL DRAFT

March 18, 2002

Foreword

This report provides answers to many questions about Network Nodes. At this point, however, additional work is needed to provide "concrete" answers to questions such as "what are the recommended Node software packages?" and "what are the Node specs?"

Throughout the remainder of 2002, several parties, including the Network Steering Board, the Network's Technical Resource Group (or its equivalent), and participants in a Beta Follow-on Project will be working to answer the many outstanding technical and institutional questions related to Nodes, as well as other Network components.

For information on the Network beyond what is enclosed in this report, see Appendix 9, "Where to Find Further Information."

Acknowledgements

Many individuals and organizations contributed to the Beta Phase project and this report. The following people offered their valuable insights and hard work:

State Participants
Tom Aten, Wisconsin Department of Natural Resources
Michael Beaulac, Michigan Department of Environmental Quality
Harry Boswell, Mississippi Department of Environmental Quality
Dennis Burling (State Co-chair), Nebraska Department of Environmental Quality
Dan Burleigh, New Hampshire Department of Environmental Services
Frank Catanese, New Hampshire Department of Environmental Services
Ken Elliot, Utah Department of Environmental Quality
Gail Jackson, Pennsylvania Department of Environmental Protection
Bill Jenks, Florida Department of Environmental Protection
Barbara Kennedy, Florida Department of Environmental Protection
Kevin Kerckhoff, Florida Department of Environmental Protection
Renee Martinez, New Mexico Environment Department
Tom McMichael, New Mexico Environment Department
Melanie Morris, Mississippi Department of Environmental Quality
Dennis Murphy, Delaware Department of Natural Resources and Environmental Control
Ben Olschewski, Utah Department of Environmental Quality
Brent Pathakis, Utah Department of Environmental Quality
Gene Pezdek, New York Department of Environmental Conservation
Chris Simmers, New Hampshire Department of Environmental Services
Mitch West, Oregon Department of Environmental Quality
John Willmott, Florida Department of Environmental Protection

EPA Participants
Chris Clark, Office of Environmental Information, Central Data Exchange
Connie Dwyer (EPA Co-chair), Office of Environmental Information, Central Data Exchange

Environmental Council of States
Kristen Dunne
Mary Blakeslee

Contractors/Consultants
Darin Black, XAware Inc.
James Boyd, Windsor Solutions
Anna Brooks, Ross & Associates Environmental Consulting, Ltd.
John Bugarin, LiveMarket
Sarah Calvillo, Ross & Associates Environmental Consulting, Ltd.
Sean Dillan, Oracle
Chenna Ganesh, Kumaran Systems
Marc Huynen, LiveMarket
Lora Jackson, LiveMarket
Derek Kishida, LiveMarket

Network Node Pilot Project – Beta Phase: Report on Project Results and Next Steps

Charles Li, Keane Federal Systems
Sugandh Mehta, IBM Global Services
Rohit Mital, XAware, Inc.
Ron Moyer, Ross & Associates Environmental Consulting, Ltd.
Mark Nobles, Logistics Management Institute
Guy Outred, Windsor Solutions
Wes Porter, LiveMarket
Ashley Rush, IBM Global Services
Louis Sweeny, Ross & Associates Environmental Consulting, Ltd.
Rob Willis, Ross & Associates Environmental Consulting, Ltd.

3/18/02

Table of Contents

The following appendices are included in a separate document:

Appendix 1. Definitions and Abbreviations of Terms

Appendix 2: Issues to be Addressed by Other Parties (Recommendations to the Board)

Appendix 3. Java Application

Appendix 4. Test Suite

Appendix 5. Node Code

Appendix 6. Example of an XML Document (Node Instance)

Appendix 7. Service Request and Response Schemas for Validation (Beta Schemas)

Appendix 8. Implementation Plan References

Appendix 9. Where to Find Further Information

List of Tables

List of Figures

Executive Summary

The objective of this report is to inform current and possible future Network Partners[1] about Network Nodes (Nodes). More precisely, this report is intended to accomplish the following:

- Explain what is (and is not yet) known about Nodes and where Node development is and should be headed during the remainder of 2002.
- Provide a proposal to the Network Steering Board (Board) for a Beta Follow-on Project for continued work starting as soon as possible.
- Explain, to the extent practicable, how to approach Node establishment for those who will be planning for and beginning to establish Nodes in early-mid 2002.

This report is *not* intended to provide background on the National Environmental Information Exchange Network (Network).

Within the general current and future Network Partner audience, this report is intended for two specific audiences: EPA and other decision makers/policy makers (such as State Chief Information Officers and program managers), and future Node implementers (those who will be either directly or indirectly working on the details of Node implementation).

EPA (representatives of Central Data Exchange (CDX)) and environmental agencies from six States (Delaware, Florida, Nebraska, New Hampshire, New Mexico, and Utah) participated in the Network Node Pilot Project – Beta Phase (Beta Phase), which began in August 2001 and ended in March 2002. Four contractors (Logistics Management Institute (LMI), XAware Inc., LiveMarket, and Ross & Associates Environmental Consulting, Ltd.) supported the project with technical advising, project coordination, and assistance with specific Node implementations.

Table ES-1 provides the Beta Phase objectives and a "quick" description of the project results relative to meeting the objectives.

---

[1] Network Partners will include EPA, States, U.S. Territories, and Tribes. In the future, the term "Network Partners" may include other government agencies and non-governmental parties.

Table ES-1. Beta Phase Objectives and Status of Completion

	Beta Phase Objective	**Status (March 2002)**[1]
1	Implement a fully normalized (i.e., rigorously standardized) facility Data Exchange Template (Template)	*Completed:* (See “Data Exchange Templates.”)
2	Involve more State agencies as appropriate	*Completed:* (Florida Dept. of Environmental Protection and the New Mexico Environment Dept.)
3	Test and resolve performance issues	*Partially Completed:* (See "Beta Node Architecture.")
4	Explore security issues and options	*Completed:* (see "Beta Node Security")
5	Link node data transfers through CDX	*In progress:* (See the discussion of the CDX Node in "Beta Node Architecture")
6	Pilot and/or Execute Trading Partner Agreements (TPAs) for additional Nodes	*N/A / not completed:* (TPA guidance was not complete in time for the project to use it as planned. See also, "Beta Phase Trading Partner Agreements")
7	Automate validation of flows against a Template	*Completed:* (See "Data Exchange Templates.")
8	Identify Node design specifications	*N/A:* Beta Phase participants decided that design specifications would not be appropriate. A Node Functional Specification, on the other hand, would be (see "Beta Node Functions.")
9	Send production data from nodes to the Facility Registry System (FRS)	*In progress:* (See the discussion of the CDX Node in "Beta Node Architecture.")
	Beta Phase "Stretch" Objectives[2]	**Status (March 2002)**[1]
1	Test interoperability (Interoperability is the ability of software and hardware on multiple machines from multiple vendors to communicate.)	*Completed* (to the extent possible) *:* (See “Beta Node Architecture” and “Beta Node Data Transport and Exchange.”)
2	Test query language options	*N/A:* Applicable query languages are still under development and using them would have been premature.
3	Scope the development of a “Node	*Completed (to the extent possible):* (See "Beta Node Architecture.")
4	Work with EPA staff to develop an EPA Node	*In progress:* (See the discussion of the CDX Node in "Beta Node Architecture.")

[1] As of mid-March 2002, a few Beta Phase participants (namely CDX) are still working on their Beta Nodes.

[2] "Stretch Objectives" were to be worked on time and resources permitting.

Beta Phase Approach (What We Did)

Beta data: The data consisted of basic facility information, such as State facility name, ID, location, environmental interest category type, and SIC code.

Table ES-2. Beta Node Toolsets and Corresponding Existing Information Systems

Beta Phase Participant	Middleware Product	Information System(s)
DNREC	BizTalk 2000	MS SQL Server 2000
NHDES	BizTalk 2000	Oracle 8i
UDEQ	BizTalk 2000	Oracle 8i
FDEP	Oracle 9iAS (with JDeveloper)	Oracle 9i
NDEQ	XAware	DB2
NMED	XAware	Oracle (TEMPO)

Beta Node requests: The Beta Phase tested three types of facility data requests for facility information: "get facility __" *By Change Date*, *By Parameter* (i.e., either by facility ID or by Environmental Interest type), and *By ID*. Nodes received these requests and responded with either the corresponding requested information or an error message.

Beta Node architecture: As shown in Table ES-2, participants used three different types of Node toolsets (based on Microsoft BizTalk, Oracle 9iAS, and XAware).

Beta Node data transport and exchange: The Beta Phase Nodes implemented two transport/exchange specifications, eXtensible Markup Language (XML) and Simple Object Access Protocol (SOAP), and experimented with a third, Web Service Description Language (WSDL).

Beta Phase Data Exchange Templates: The Beta Phase used Templates (Schemas) to structure and express all messages processed by Nodes. Additionally, Beta Phase participants used the Schemas to validate the service requests and responses (see "Data Exchange Templates"). Throughout most of the project, participants intended to use the Schemas to validate the service response that corresponded with the *By Change Date* and *By ID* request types. However, several iterative attempts revealed complex issues (with unclear and complex solutions) with how these Schemas had been designed. It was not possible to validate against the given Schemas without substantially extending project duration.[2] However, the Beta Phase successfully used a Schema to validate the *By Parameter* service request type. Further, participants used the prototype Network Registry/Repository to store and retrieve the most current Schemas.

Beta Node Performance: The Beta Phase included two Node performance evaluations, one that tested the speed of the Node processing time and one that tested the speed of the total transaction time (from the moment the request for information is sent to when a response is received by the requestor).

[2] In early 2002, participants communicated these issues to the Network's Technical Resource Group (TRG), which has already begun discussions on how to resolve them. The TRG (or perhaps another party designated by the Board) will develop recommendations on issues like these for future Network Template development.

Beta Nodes – The Technical Side: Results and Lessons Learned

The technical side of Beta Nodes includes several interrelated subject areas, including Node functions, Node architecture and specifications, Node data transport and exchange, Data Exchange Templates (Schemas in the Beta Phase), connecting to existing information systems, and Node performance. Figure ES-3 (on following page) provides a high-level picture of these Nodes components and the relationships between these components and the local technical environment of Network Partners.

Beta Node Functions: High-level Results and Lessons Learned

Like the Network, Nodes are defined by the functions they perform. Node functions fall into three categories: functions performed while requesting information (requestor Node functions), functions performed while fulfilling information requests (requested Node functions), and general Node management functions. The Beta Phase focused on requested Node functions (see Figure ES-4). A simple Java test application performed the requestor Node functions for testing purposes.

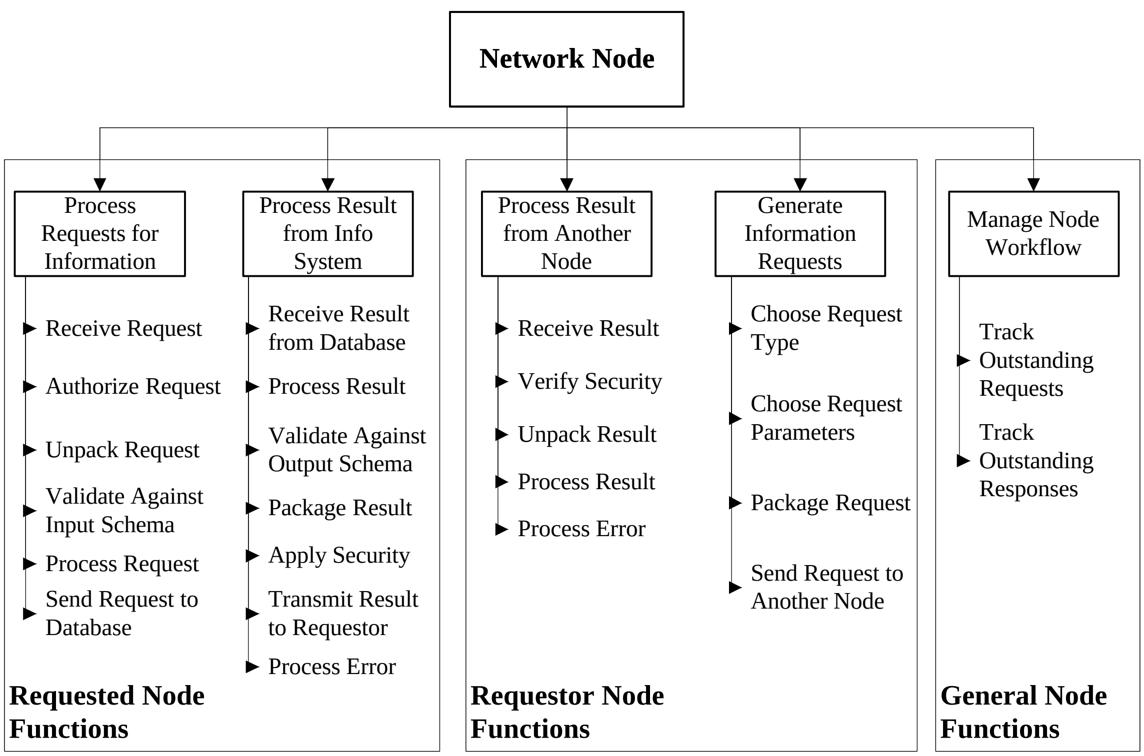
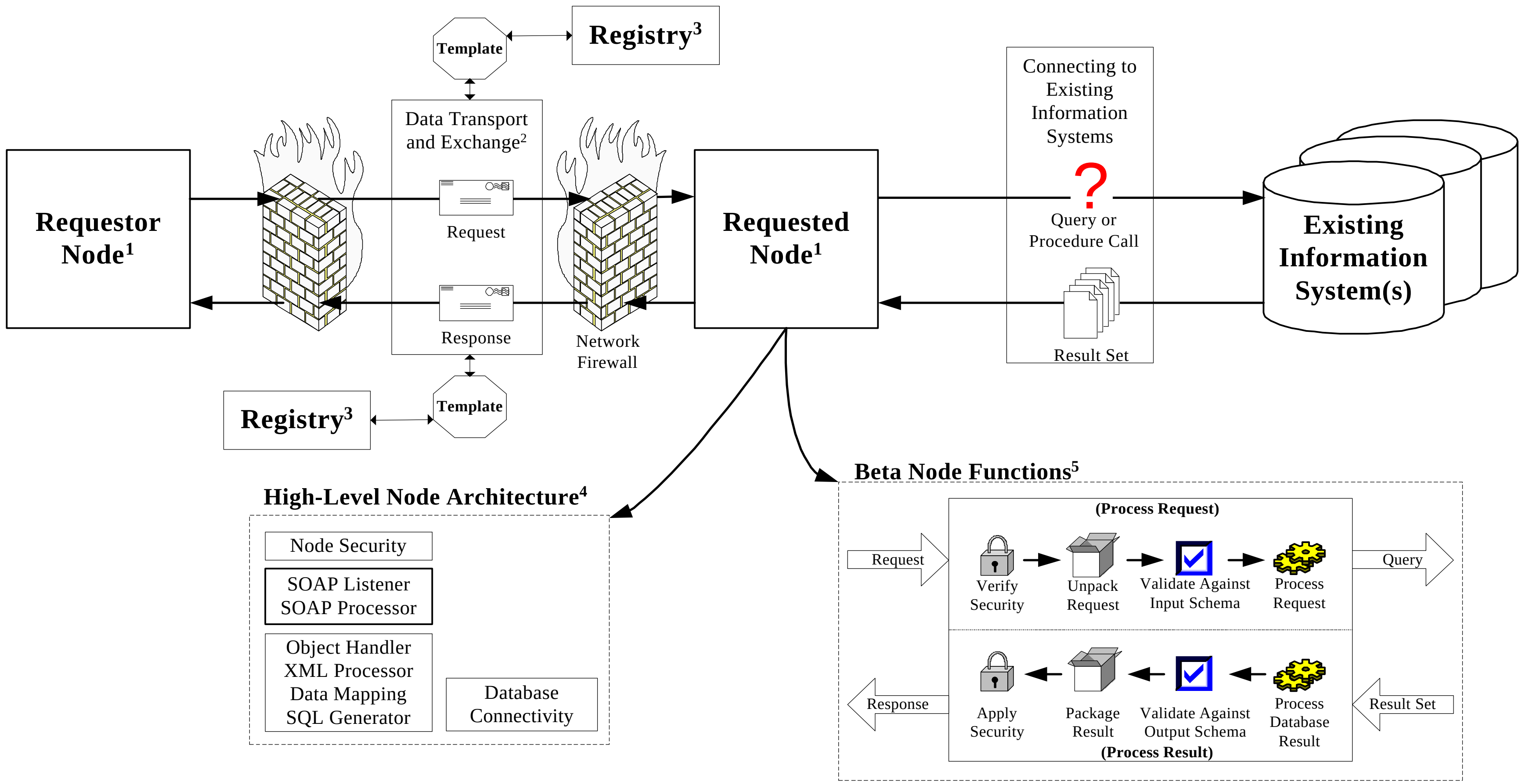


Figure ES-4. Beta Node and Anticipated Node Functions



Notes:
[1] - Requestor Node functions include making requests and processing responses from requests. Requested Node functions include processing incoming requests and processing results sets from existing information systems.
[2] - The Board will develop exchange protocols and implementation standards. This will likely include WSDL.
[3] - Schema (Template) validation could occur up to four times for each transaction. The Beta Phase participants recommend that the Board further investigate Template validation.
[4] - Technical Specifications for each Partner's Node will be based on tool choices and their own computing environment. They will be developed by each Partner.
[5] - Functional Specifications will be developed by the Network Steering Board and shared with all Partners.

Figure ES-3. Network Nodes and Their Components

The "steep learning curve" experienced by Beta Phase participants when determining Beta Node functions led the group to recommend the developing a Network Node Functional Specification. Although it is likely that the high-level Node functions outlined in Figure ES-4 are likely to be the "correct" set, specific areas still require further investigation, including generating service requests and determining the optimal place and timing of many functions such as security and validation. The Node Functional Specification must explicitly define each function to maximize Network interoperability and efficiency while allowing Node implementation flexibility.

Beta Node Architecture and Specifications: Results and Lessons Learned

The basic architecture for all Nodes is the same because all Nodes have to perform the same high-level functions. Each Node must have an Internet connection, be hosted on a server, and connect to an information system (see Figure ES-5). However, the specific architecture of each Node will differ depending on the specific Node toolsets (hardware and software) and configurations. In the Beta Phase, there were no dramatic differences in the "performance" of the three toolsets used (BizTalk, Oracle, and XAware), even though each State agency implemented its Node differently.

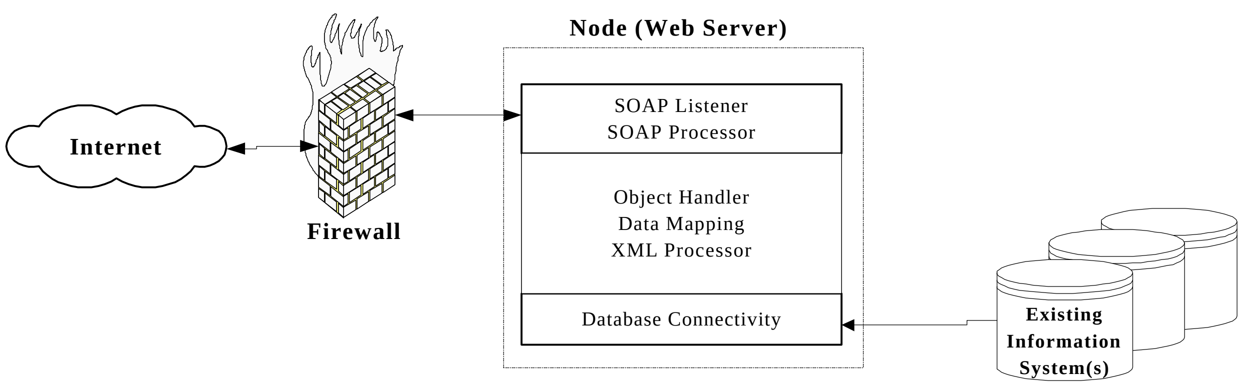


Figure ES-5. Basic Node Architecture

The Beta Phase experience indicates that local considerations will be the basis for Node architecture decisions. In some cases, Partners will be able to implement Nodes using combinations of their existing hardware and software plus some in-house expertise. The Beta Phase implemented only three of the many available Node toolsets. Although each Beta toolset operates as envisioned, none can be strongly recommended or eliminated based on the Beta Phase experience. They all work. Further evaluation is needed to determine whether one or two are significantly more/less well suited for Nodes.

Beta Node Data Transport and Exchange: High-level Results and Lessons Learned

The Beta Phase Nodes implemented two transport/exchange mechanisms, eXtensible Markup Language (XML) and Simple Object Access Protocol (SOAP), and experimented with a third, Web

Services Description Language (WSDL). Participants used SOAP to define a standard "electronic envelope" for the transport of the XML requests and data. Although available software supports SOAP and SOAP can (and arguably should) function as the Network's basic transport/exchange standard, appropriate Network usage of the W3C SOAP Specification must be defined to ensure consistent Node implementation and interoperability. Also, as with any new standard, Partners should be skeptical of vendors' "compatibility" claims.

Data exchanged between the Florida and CDX nodes demonstrated the use of WSDL to provide automated inquiries of remote Nodes' capabilities and exchange requirements. WSDL can be used to document quickly and efficiently transport and exchange requirements, configure Nodes for Flows, and create applications that retrieve data from Nodes. Future Partners will need direction on the appropriate usage WSDL (in the yet-to-be developed Network Exchange Protocol and the Node Functional Specification).

Connecting To Existing Information Systems: Results and Lessons Learned

Most Beta Phase participants spent a majority of their Node development time connecting Nodes to their information systems. Data mapping from the information systems to the Schemas required careful implementation, a combination of technical and programmatic expertise, and close communication between participants' Node teams and agency data adminis
news" is that existing application integration tools work for connecting Nodes to a wide variety of existing information systems. However, Network Partners will benefit from recommendations on the most effective ways to query data and how to support flexible querying (e.g., ad hoc queries).

Beta Node Performance: High-level Results and Lessons Learned

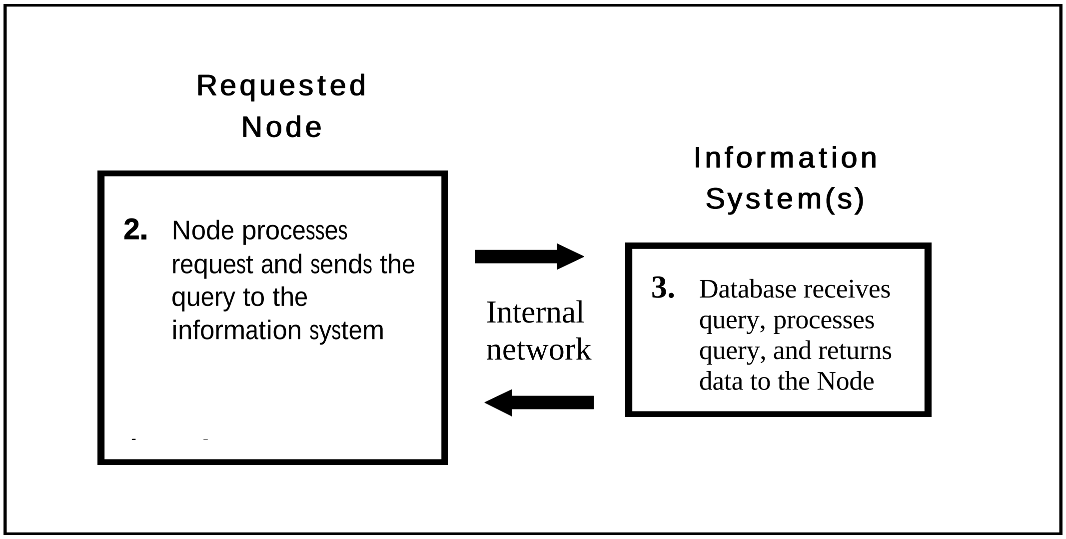
The Beta Phase included two Node performance evaluations:

<u>Performance that measured the total transaction time.</u> (See Figure ES-6.) The total transaction time has five steps:

1. The requestor Node (or, in the case of the Beta Phase, the Java test application) sends a service request to the requested Node.
2. The requested Node processes the service request and sends the resulting query to the information system.
3. The information system processes the query and sends the data back to the requested Node.
4. The requested Node processes the data and sends the service response back to the requestor Node.

5. The requestor Node receives the service response.

Figure ES-6. Steps Included in Node Processing Time



Performance that measured the Node processing time. (See Figure ES-7.) The Node processing time is a subset of the total transaction time (described above). The Node processing time *begins* when the requested Node receives the service request and *ends* when requested Node sends the service response.

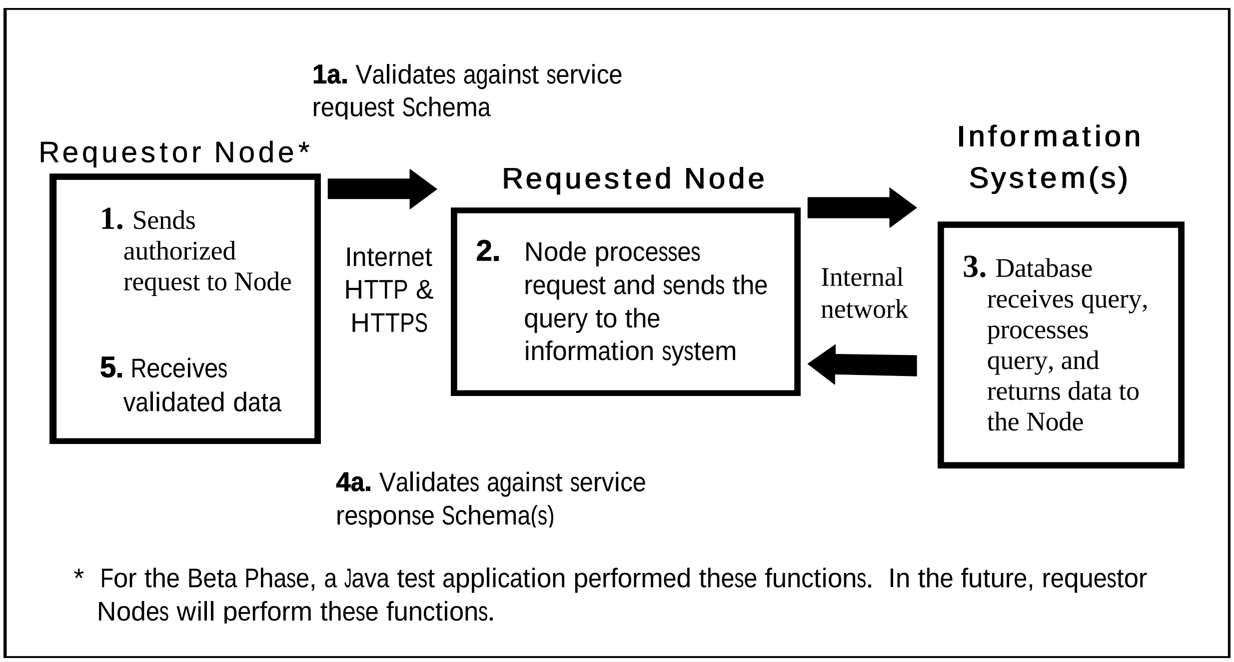


Figure ES-7. Steps Included in Total Transaction Time

Lessons Learned

From high-speed Internet connections, the total transaction times were acceptable: the smallest requests (for one specific record) took one second or less. Transactions of a few megabytes took between 30 seconds and a few minutes. Within the total response time, the Node processing time (i.e., not the Internet time) dominated the total transaction time. It is important to realize that while the speed of the Network is important, an implicit design assumption of the Network is that current batch transactions that produce large data sets could be accommodated by increasing the frequency of transactions thereby reducing both delays and file sizes. Obviously, the transaction requirements of some Flows may not be amenable to this approach. In those cases, other performance solutions may include compression or alternative transport protocols.

The secure (HTTPS) transaction times were slower by approximately 20 percent. (Note that these were exchanges between a HTTP request and HTTPS response.) External (non-Node) factors that also affected performance included the following: Internet connection speed (this is likely to be the largest performance variable), network type, existing system congestion, existing system hardware, database structure and tuning, and query method. Finally, how Partners' choose to query their information systems can affect performance.

The following sections cover these technical topics in more detail.

Beta Node Planning and Management: Lessons from the Field

Node establishment, like other information technology projects, requires a coordinated approach and development methodology. The factors behind project ease or difficulty depend on resource availability, in-house expertise, and external influences that are out of the project staff's control. Establishing a Node is no different.

The Beta Phase demonstrated that States can build Nodes now. The Beta Phase lessons learned and recommendations, coupled with the (recommended) work to be done in the Beta Follow-on Project, will help make Node establishment easier for future Node builders. However, even if a perfect "how to" Node guide were provided to all Partners prior to Node establishment, there would always be the potential for any number of unforeseen challenges to arise.

The Beta Phase experience suggests that Node establishment requires the same type of planning required by any application integration effort.

The Beta Phase high-level lessons learned regarding Node establishment also apply to most application integration efforts. They are as follows:

- Establishing a Node will be easier once the Node Functional Specification and Network Exchange Protocol have been developed (and tested and are ready for use). The sooner these are ready, the better for overall Network Implementation.
- The staff establishing the Node (or maintaining/managing after initial establishment) need to have some understanding of Network technologies.
- Support (e.g., resources and authorization/clearance) for the Node from senior decision-makers and managers is important for timely and efficient Node implementation.

- Key staff, who are often located in different areas, will need to coordinate to support Node implementation.
- Outside technical support is likely to be needed for initial Node establishment.
- Given the newness of some of these technologies, outside support may be difficult to find at first.
- There is no such thing as one Demonstrated Node Configuration that "fits all," and every Partner will have more than one suitable configuration. Similarly, no Node configurations used to date have been found unsuitable.

Planning for Node Costs

There is no simple formula for Node costs. Each Partner's unique needs dictate and interests will determine most if not all Node cost factors. The costs of establishing the Beta Nodes do not represent the expected costs for future Nodes and are therefore not repeated here. This Executive Summary does not provide separate cost information because the topic of Node costs should not be oversimplified. For a more detailed discussion, see "Estimating Node Costs"

Beta Phase: Trading Partner Agreements (TPAs)

Initially, the Beta Phase intended to use the expected official Network TPA guidance (that was expected to be complete in the fall of 2001) to develop new TPAs for the Beta Phase. But the TPA development guidance was delayed – the Board is now expected to issue guidelines and checklists in quarter four of 2002. This prompted the Beta Phase participants to put the TPA objective on hold, and ultimately, participants agreed that signing additional TPAs was not as important as other project objectives.

Building on the Beta: A Proposal to the Network Steering Board for a Follow-on Project

Beta Phase participants recommend that the Board immediately commission and launch a three-phased Beta Follow-on Project. The project would have the following primary objectives (each driving one project phase):

1. Develop the draft Node Functional Specification (Functional Specification) and the draft Network Exchange Protocol (Exchange Protocol).

2. Test the Functional Specification/Exchange Protocol in controlled Node implementations/Flows using a small test group.

3. Use these tests to revise and document working versions of the Functional Specification and Exchange Protocol for general use.

For a more complete description of the Beta Follow-on Project, please refer to "Building on the Beta: A Proposal to the Network Steering Board for a Follow-on Project." The proposal concludes the main section of this report.

I. Introduction

A. Report Objectives

The objective of this report is to inform those interested in Network Nodes (Nodes)[3] (see "Report Audience" below). More specifically, this report is intended to:

- Explain what is (and is not yet) known about Nodes, based on the experience of the Network Node Pilot Project – Beta Phase (Beta Phase) participants.
- Explain where Node development is and should be headed during the remainder of 2002. Note that the *Network Implementation Plan (Implementation Plan)* outlines a general schedule for the development of Node Functional Specification (Functional Specification)[4] and a (Node-related) Network Exchange Protocol (Exchange Protocol)[5] later in 2002.
- Provide a proposal to the Network Steering Board (Board) for a Beta Follow-on Project for continued work starting in April 2002.
- Explain, to the extent practicable, how to approach Node establishment for those who will be planning for and beginning to establish Nodes in early-mid 2002.

This report is *not* intended to provide background on the National Environmental Information Exchange Network (Network), of which Nodes are a key component. Most of this report assumes a familiarity with the Network and its basic technical framework. (For readers who are not familiar with the terms Data Exchange Template (Template), Trading Partner Agreement (TPA),

---

[3] **Node or Network Node** – A set of tools to exchange information on the Network. A node uses the Internet, a set of standard protocols, and appropriate security measures to issue and respond to authorized requests for specific information. A Node is a simple environmental web service that initiates requests for information, processes authorized queries, and sends the requested information in a standard format (XML). A Node also validates this information against a pre-defined Schema or Data Exchange Template (DET).

[4] **Node Functional Specification** – the detailed description of a Node's expected behavior. This description will include the functions the Node will perform, how those functions are invoked and the output expected. The specification may also include a limited number of internal Node behaviors (e.g. logging and security). Since most Node functions involve data exchange, the Node Functional Specification will rely heavily on the Network Exchange Protocol to define the types of valid messages a Node should expect and be able to produce. The Node Functional Specification may be supported by a standard "test suite" of service requests and responses that Nodes must be able to process.

[5] **Network Exchange Protocol** – the set of rules that governs the generation and use of valid service requests and responses. Since the Network is based on pre-existing standards, the Network Exchange Protocol will provide addition rules that govern how those standards (namely XML, XML Schema, SOAP, and WSDL) are implemented on the Network. These additional rules are needed because these standards, given their extensibility, can be implemented in many operational but mutually incompatible ways. Network Exchange Protocols will be used by implementers to take their data content standards (defined in Templates) and embed them in valid service requests and responses. They can also be used to confirm or establish the validity of one's own or a Partners service requests and responses.

eXtensible Markup Language (XML)[6], and the Board, additional background reading is recommended.) The "Foreword" in this report directs readers to several other sources of information about the Network.

B. Report Audience

This report is written for one general audience and, within that general audience, two specific audiences. The general audience is those (States, Tribes, Territories, Environmental Protection Agency (EPA), and other parties) who will be participating in the Network. Specifically, this report is intended for two audiences: EPA and State decision makers/policy makers (such as State Chief Information Officers and Program Managers), and future Node implementers (those who will be either directly or indirectly working on the details of Node implementation). Note that some repetition in subject matter was necessary to achieve these report objectives.

The "Executive Summary" and "High-level" sections of this report are intended for decision makers/policy makers, while the more detailed sections are geared more toward the Node implementers. Two sections are intended specifically for the Board: "Building on the Beta: A Proposal to the Network Steering Board for a Beta Follow-on Project," and "Appendix 2: Issues to be Addressed by Other Parties (Recommendations to the Board)."

C. Report Organization

Corresponding Network Implementation Plan Milestone:

Based upon the completion and recommendations of the Beta Phase, the Board will establish a Beta Phase "follow-on" effort in early 2002.

(*"Bringing the Pieces Together: Continuation of Network Implementation Pilots"*)

This report has three major sections:

1. *Beta Phase Nodes (Beta Nodes): the Technical Side*

   In addition to describing what Nodes are, this section details the major technical aspects of a Node (Node functions, architecture, data transport and data exchange), as well as the Node-related technical issues (Templates), connecting to existing information systems, and Beta Node performance).

2. *Beta Node Planning and Management: Lessons from the Field*

   This section outlines the lessons learned about how to approach Node establishment and use (i.e., Network participation). Not only do Nodes need to be designed carefully, the Node-related components of the Network (e.g., Template testing/development, and connecting to existing information systems) also need to be included in this planning process and carefully managed.

[6] **XML** – eXtensible Markup Language is a markup language defined by the W3C that provides a strict set of standards for document syntax while allowing developers, organizations, and communities to define their own vocabularies.

3. *Next Steps and Recommendations for Follow-on Work*

   This section begins with a high-level overview of the work that is still needed to create the Functional Specifications by the fourth quarter of 2002. Most of the recommendations focus on the need for a "Beta Follow-on Project" that will build upon the existing Node knowledge (captured in this report). This report concludes with a proposal from the Beta Phase participants to the Board for this Follow-on work.

D. Background on the Network Node Pilot Project

In January 2001, the Interim Network Steering Group (INSG)[7], together with the State and EPA Co-chairs of Network-related Action Teams and the Environmental Data Standards Council (EDSC), decided to produce near-term, visible results that would demonstrate the advantages of the Network approach. To produce these results, an ad-hoc group was formed to conduct the Network Node Pilot Project (Pilot Project). The Pilot Project's general goals were as follows:

- To prove the XML technology concept referenced in the *Blueprint for a National Environmental Information Exchange Network (Blueprint)*
- To demonstrate that State agencies have the capacity to develop Nodes and to demonstrate the use of XML transfers and integration
- To establish a connection and exchange data between States' Nodes and EPA's Central Data Exchange (CDX)
- To establish Nodes that concur with formal TPAs
- To validate the Node transactions against registered Templates that normalize data into a single common format
- To develop recommendations for Node coordination

To date, two Phases (Alpha and Beta) of the Pilot Project have been completed.

Alpha Phase: Background

Representatives of four State environmental agencies [Delaware Department of Natural Resources and Environmental Control (DNREC), New Hampshire Department of Environmental Services (NHDES), Nebraska Department of Environmental Quality (NDEQ), and Utah Department of Environmental Quality (UDEQ)] initiated the Alpha Phase in the spring of 2001. These agencies used their own resources, background, and experience with Network development [Small States Technical Assistance Initiative, Facility Data Action Team (FDAT), Facility Identification Template for States Version2 (FITS II), and Facility Registry System (FRS) pilots] to establish the first Node data exchanges between information requestors and State agencies' existing information systems. Although EPA did not directly participate in the Alpha Phase, it provided support for contractor coordination and technical Node implementation support. The project was successful; not only

[7] The charter of the INSG sunsetted in February 2002. As of February 2002, the Board has assumed the INSG's responsibilities.

did it produce the first "proof of concept" Network Nodes, it also provided a clear understanding that further pilot work was necessary. Topics for further evaluation included the need to define an approach for querying Nodes, demonstrating inter-operability, testing performance, and exploring security options. These and other issues are documented in the Alpha Phase results report (See "Appendix 9: Where to Find Additional Information.") Based on these issues, the Alpha Phase participants made recommendations for the Beta Phase in their final report.

Beta Phase: Background

In June 2001, the Beta Phase began when the INSG approved a proposal by the Alpha Phase participants for the Beta Phase of the Pilot Project. Six State agencies [the four Alpha Phase agencies plus Florida Department of Environmental Protection (FDEP) and New Mexico Environment Department (NMED)] and EPA (representatives of CDX) participated in the Beta Phase, which began in August 2001 and finished in February 2002. Four contractors [Logistics Management Institute (LMI), XAware, Inc., LiveMarket, and Ross & Associates Environmental Consulting, Ltd.] supported the project with technical advising, project coordination, and assistance with specific Node implementations.

Table 1 below provides the Beta Phase objectives and a "quick" description of the project results relative to meeting the objectives.

Table 1. Beta Phase Objectives and Status of Completion

	Beta Phase Objective	Status (March 2002)[1]
1	Implement a fully normalized (i.e., rigorously standardized) facility Data Exchange Template (Template)	*Completed:* (See "Data Exchange Templates.")
2	Involve more State agencies as appropriate	*Completed:* (Florida Dept. of Environmental Protection and the New Mexico Environment Dept.)
3	Test and resolve performance issues	*Partially Completed:* (See "Beta Node Architecture.")
4	Explore security issues and options	*Completed:* (see "Beta Node Security")
5	Link node data transfers through CDX	*In progress:* (See the discussion of the CDX Node in "Beta Node Architecture")
6	Pilot and/or Execute Trading Partner Agreements (TPAs) for additional Nodes	*N/A / not completed:* (TPA guidance was not complete in time for the project to use it as planned. See also, "Beta Phase Trading Partner Agreements")
7	Automate validation of flows against a Template	*Completed:* (See "Data Exchange Templates.")
8	Identify Node design specifications	*N/A:* Beta Phase participants decided that design specifications would not be appropriate. A Node Functional Specification, on the other hand, would be (see "Beta Node Functions.")

9	Send production data from nodes to the Facility Registry System (FRS)	*In progress:* (See the discussion of the CDX Node in "Beta Node Architecture.")
Beta Phase "Stretch" Objectives[2]		**Status (March 2002)**[1]
1	Test interoperability (Interoperability is the ability of software and hardware on multiple machines from multiple vendors to communicate.)	*Completed* (to the extent possible) *:* (See "Beta Node Architecture" and "Beta Node Data Transport and Exchange.")
2	Test query language options	*N/A:* Applicable query languages are still under development and using them would have been premature.
3	Scope the development of a "Node	*Completed (to the extent possible):* (See "Beta Node Architecture.")
4	Work with EPA staff to develop an EPA Node	*In progress:* (See the discussion of the CDX Node in "Beta Node Architecture.")

[1] As of mid-March 2002, a few Beta Phase participants (namely CDX) are still working on their Beta Nodes.
[2] "Stretch Objectives" were to be worked on time and resources permitting.

Beta Phase Approach (What We Did)

The general Beta Phase approach was as follows:

Between June and August 2001, the Alpha Phase participants (who had already decided to participate in the Beta Phase) examined a series of questions, such as which additional States to ask to participate in the Beta Phase, what data to exchange, what Templates to use, etc. The decisions based on these questions are reflected in the approach areas described below:

Participation of New State Agencies

> **Corresponding Network Implementation Plan Milestone:**
>
> In 2002, Partners will develop functional Node prototypes for basic flows in Six Beta Phase States.
>
> (*"Bringing the Pieces Together: Continuation of Network Implementation Pilots"*)

The existing participants invited FDEP and NMED to participate because these agencies use Oracle for most (or all) of their information management systems, and Oracle products are used by many potential Network Partners (Partners) (if not a majority of State environmental agencies). The specific interest in FDEP stemmed from the fact that the agency is from a "large" State, and only "small" State agencies had been involved to that point. Participants were specifically interested in NMED because it uses TEMPO (a customized, proprietary, Oracle-based application that several other State agencies also use) as its primary information system. Further, agency staff in both FDEP and NMED were already involved in Network development efforts.

For similar reasons, the participants also invited two other State agencies to participate, Pennsylvania Department of Environmental Quality and Mississippi Department of Environmental

Quality. For different reasons, both these agencies decided against inclusion in the Beta Phase, but they have continued to be part of the broader Beta Phase discussions.

The Beta Phase involved six different hardware and software configurations, all of which successfully exchanged information. See “Beta Node Architecture” for a detailed description of the Participants’ configurations. As shown in Table 2, each State agency differed in the extent of Node Implementation.

Table 2. Beta Node Implementations Differences

State Agency	Port 80	Port 443 (SSL)	Request *By Change Date*	Request *By Parameter*	Request *By ID*	Zipped Files	Validation (*By Parameter*)
DNREC	Y	Y	Y	Y	Y	N	Y
FDEP*	Y	N	N	Y	N	N	Y
NDEQ	Y	N	Y	Y	Y	Y	Y
NHDES	Y	Y	Y	Y	Y	N	Y
NMED	Y	P	N	Y	N	N	Y
UDEQ	Y	Y	Y	Y	Y	N	Y

Y=Yes, N=No, P=Partial

* FDEP implemented three different Node implementations. (See “Beta Node Data Transport and Exchange”)

The fact that not all participants implemented all possible Beta Phase options should not be over-interpreted. Given the limited time and staff resources available, Beta Phase participants sought to be strategic about their investments to ensure that key functions were implemented by as many participants as possible and that all options were implemented by at least one participant.

State Agency	Middleware Product	Information System(s)
DNREC	BizTalk 2000	MS SQL Server 2000
NHDES	BizTalk 2000	Oracle 8i
UDEQ	BizTalk 2000	Oracle 8i
FDEP	Oracle 9iAS (with JDeveloper)	Oracle 9i
NDEQ	XAware	DB2
NMED	XAware	TEMPO

Table 3. Beta Node Toolsets and Corresponding Existing Information Systems

Beta Data

The "Beta data" consisted of basic facility information, such as State facility name, ID, location, environmental interest category type, and SIC Code[8]. Facility information is typically related to many other kinds of data collected by State environmental agencies, and as such, it tends to be a data "centerpiece" of most State data models. EPA does not currently require States to submit facility data, but is increasingly interested in collecting facility data through the FRS. EPA is working with State agencies to explore the best uses of facility data for a variety of important purposes. For these reasons, and because learning how to share facility data would be a challenging, but useful, benchmark for a pilot Network Flow (Flow), the Beta Phase participants chose to continue using facility data for the Beta Phase (as in the Alpha Phase).

Beta Phase Service Requests

The Beta Phase tested three Node services requests, informally called "get facility…" 1) *By Change Date*, 2) *By Parameter*, and 3) *By ID*. Each of these requests returned facility records by these parameters. (See "Beta Node Functions"). Furthermore, Beta Phase participants could select certain values, such as the Environmental Interest Type, to make the request more specific and detailed. The Beta Phase used a Java test application to customize these service requests. (See "Appendix 3: Java Application")

Beta Node Architecture

The Beta Phase participants approached Beta Node architecture with three Node technology technical toolsets: Microsoft BizTalk 2000, Oracle 9i Application Server (with a JDeveloper add-on), or XAware XA-iServer. The details of each participant's implementation varied to fit the specific needs and computing environments. Participants also chose to use somewhat different data subsets based on their needs for data security, access issues with network firewalls, and the structure of their existing information systems/databases.

Beta Node Data Transport and Exchange

Beta Nodes implemented two transport/exchange standards – eXtensible Markup Language (XML) and Simple Object Access Protocol (SOAP)[9] – experimented with a third, Web Service Description Language (WSDL)[10]. SOAP is becoming the de-facto standard for XML data transport on the Internet. However, because the SOAP Standard is still evolving, even the simple Beta Phase SOAP implementation demonstrated the need for further clarification on Network use of SOAP.

---

[8] **SIC Codes** – Standard Industrial Classification Codes are numerical codes designed to create uniform descriptions of business establishments.

[9] **SOAP** – Simple Object Access Protocol is an XML/HTTP-based protocol for accessing services, objects and servers in a platform-independent manner.

[10] **WSDL** – Web Services Description Language defines the beginning and end point of a service that allows other computers to access and invoke its function. WSDL provides other computers the structure to determine what a web service does, what a web service needs to work, and how to invoke it.

Beta Phase Data Exchange Templates

The Beta Phase used Templates (Schemas[11]) to structure and express all messages processed by Nodes. Additionally, Beta Phase participants used the Schemas to validate the service requests and responses (see "Data Exchange Templates"). Though the Beta Phase intended to use Schemas for the validation of service response *By Change Date* and *By ID*, participants decided against this because initial validation revealed complex issues (with unclear and complex solutions) in how the Schemas had been designed. However, the Beta Phase successfully validated for service request *By Parameter*. Further, participants used the prototype Network Registry/Repository to store and retrieve the most current Schemas.

Beta Node Performance

The Beta Phase included two Beta Node performance evaluations, one that measured the speed of the total transaction time (i.e., the time that passed from when a request was sent and a response was received) and one that tested the speed of just the Node processing (listening for and processing requests/responses. To capture performance data in both these areas, a time-stamping feature was built into the Java test application that determined the time required for each stage of processing the service request.

The more detailed sections throughout this report further describe the Beta Phase approach.

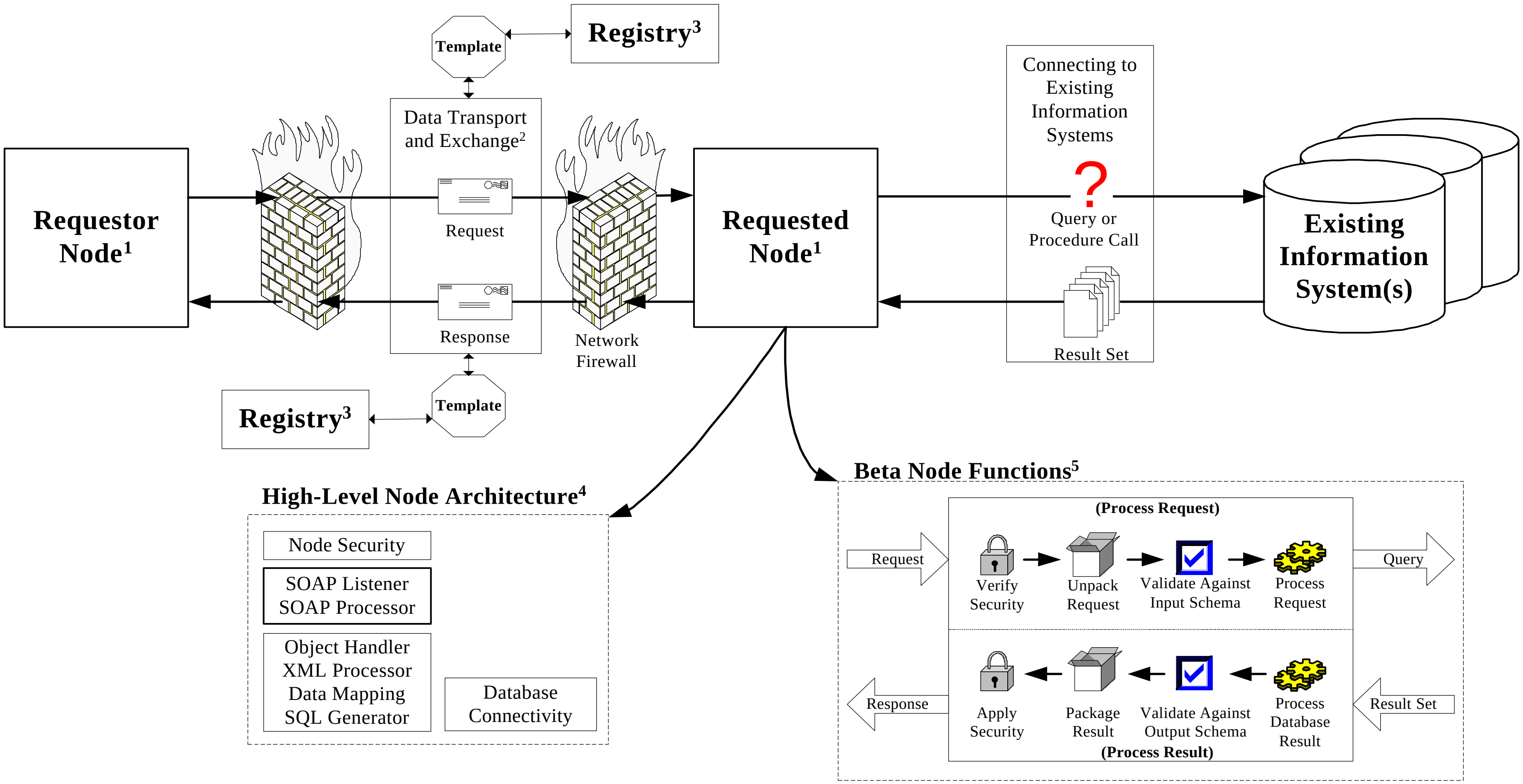
[11] **Schema** – A database-inspired method for specifying constraints on XML documents using an XML-based language. Schemas address deficiencies in DTDs, such as the inability to put constraints on the kinds of data that can occur in a particular field (for example, all numeric). Schemas are hierarchical and can create an unambiguous specification. They can also determine the scope over which a comment is meant to apply.

II. Beta Phase Nodes: The Technical Side

E. High-level Overview and Lessons Learned

The technical side of Beta Nodes includes several interrelated subject areas, including Node functions, Node architecture and specifications, Node data transport and exchange, Data Exchange Templates, connecting to existing information systems, and Node performance. Figure 1 provides a high-level picture of these Node components and their local technical environments.

The next few pages provide high-level overviews and associated lessons learned for each major technical area. Technical readers who would prefer a more detailed discussion may want to skip this overview section.



Notes:
[1] - Requestor Node functions include making requests and processing responses from requests. Requested Node functions include processing incoming requests and processing results sets from existing information systems.
[2] - The Board will develop exchange protocols and implementation standards. This will likely include WSDL.
[3] - Schema (Template) validation could occur up to four times for each transaction. The Beta Phase participants recommend that the Board further investigate Template validation.
[4] - Technical Specifications for each Partner's Node will be based on tool choices and their own computing environment. They will be developed by each Partner.
[5] - Functional Specifications will be developed by the Network Steering Board and shared with all Partners.

Figure 1. Network Nodes and Their Components

Beta Node Functions (High-level Overview and Lessons Learned)

Overview

Like the Network, a Node is defined by the functions it performs. Node functions fall into three categories: 1) functions performed while requesting information (requestor Node functions), 2) functions performed while fulfilling information requests (requested Node functions), and 3) general Node management functions. The Beta Phase focused on requested Node functions (see Figure 2), which, at a high level, are processing requests for information and processing results from the information system(s)/database(s). For the Beta Phase, a simple Java test application performed the requestor Node functions for testing purposes. (See Appendix 3: "Java Test Application.")

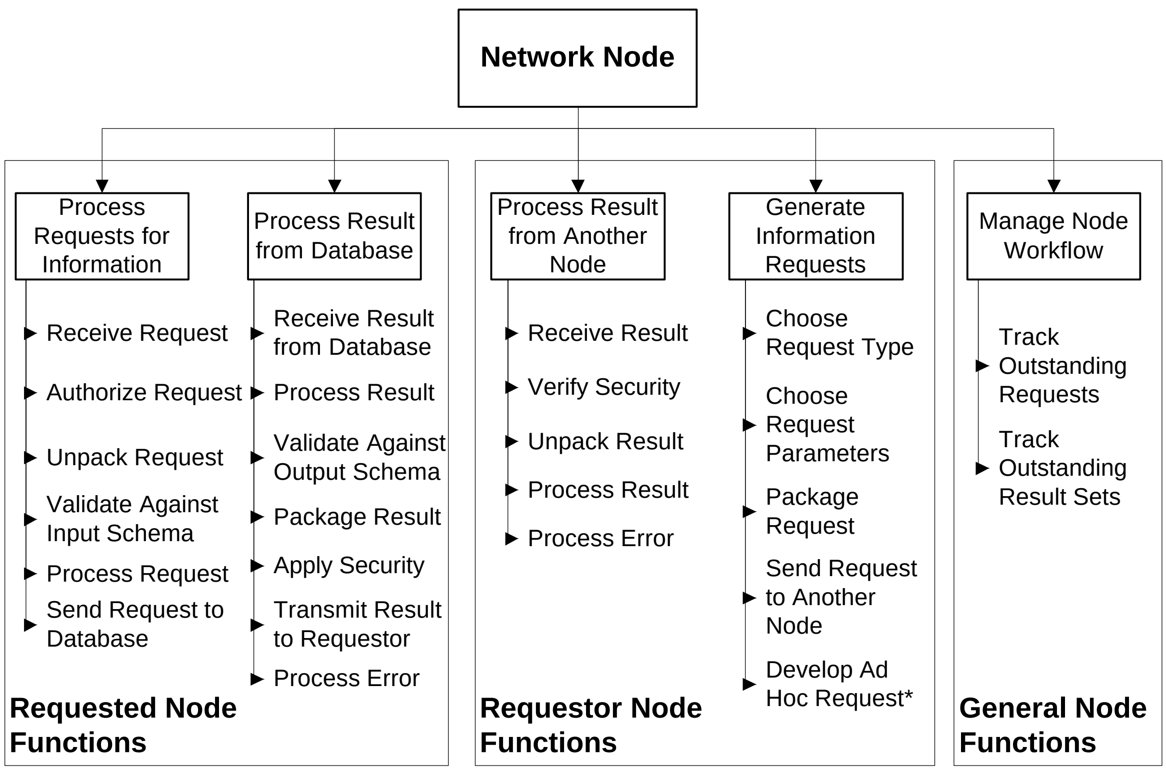


Figure 2. Beta Node and Anticipated Node Functions

Lessons Learned

To further the Network, the Beta Phase helped to define both the scope and operation of Nodes as Nodes were being built. Participants experienced a "steep lea
the appropriate Node functions. Though the functions outlined in Figure 2 are likely to be the "correct" set, participants strongly recommend that further work be done in this area as soon as possible. This work will lead to a Network Node Functional Specification that will enable Partners to know what the functional expectations are for establishing their Nodes. (See "High-Level Next Steps and Recommendations"). Specific areas for further investigation include how best to

generate service requests and determining the optimal place and timing of many functions such as security and validation. The Node Functional Specification must explicitly define each function to maximize consistency of the Network while allowing for flexibility in Node implementation.

> **Corresponding Network Implementation Plan Milestone**:
>
> The Board is prepared to produce Node Functional Specification every six months, starting in Fall 2002.
>
> (*"Establishing Network Nodes"*)

Beta Node Architecture (High-level Overview and Lessons Learned)

Overview

The basic architecture for all Nodes is the same because all Nodes have to perform the same high-level functions (See Figure 3). Each Node must have an Internet connection, be hosted on a server, and connect to an information system. However, the specific architecture of each Node will differ depending on the specific Node tools (hardware and software) and configuration. In the Beta Phase, there were no dramatic differences in the "performance" of the three toolsets used (Biztalk, Oracle, and XAware), even though each State agency implemented its Node differently.

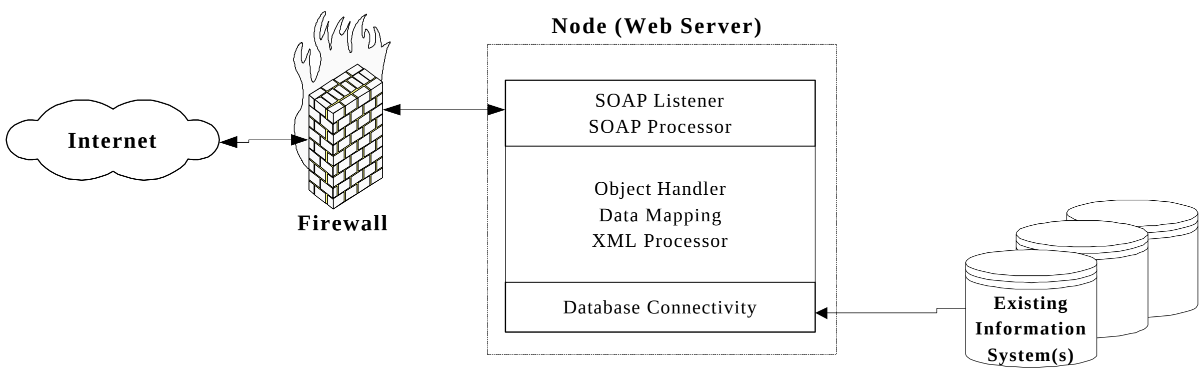


Figure 3. Basic Node Architecture

Lessons Learned

The Beta Phase experience indicates that local considerations (e.g., specific add-on functions that a Partner would like its Node to perform or the compatibility with the existing systems and a particular Node tool) will be the basis for Node architecture decisions. In some cases, Partners will be able to implement Nodes using combinations of their existing hardware and software plus some in-house expertise. For example, the Beta Phase participants in the UDEQ think that a fully-functional Node will not require more than 25 percent capacity of their existing Biztalk server (the "excess" capacity would continue to perform other functions such as e-commerce).

Participants agree that, based on the Node-building experience to date, none of the three Beta Phase Node tools (Biztalk, Oracle, and XAware) can be either strongly recommended or eliminated. They all work. Further evaluation is needed to determine whether one or two are significantly more/less well suited for Nodes.[12]

Beta Node Data Transport and Exchange Process (High-level Overview and Lessons Learned)

Overview

The Beta Phase Nodes implemented two transport/exchange mechanisms (XML and SOAP) and experimented with a third (WSDL). Because expectations for specific Node data transport and exchanges were not pre-defined, SOAP implementation was directed by the kind of hardware and software used. The different SOAP implementations negatively impacted Node-to-Node interoperability. Seamless exchanges were only possible once several "tweaks" were made to insure consistent implementation.[13] The Network's ability to provide many services to many Partners depends upon the interoperability of the products used to implement Nodes.

> **Corresponding Network Implementation Plan Milestone:**
>
> New versions of the Network Exchange Protocol are scheduled to be released at the end of every calendar year.
>
> (*"Refining the Details of the End-to-End Information Exchange Process"*)

Lessons Learned

Both the Node Functional Specification and the Network Exchange Protocol (these two will be closely related) are needed to clarify the answers to these types of questions.

SOAP is (mostly) supported by available software and can function as a basic transport/exchange standard. However, consistent Network application of the SOAP Specification must be defined to ensure Node interoperability. Also, as with any new standard (W3C), Partners should be skeptical about "compatibility" of vendors.

WSDL can be used to document quickly and efficiently transport and exchange requirements, configure a Node for Flows, and create applications that retrieve data from Nodes. The Node Functional Specification and Network Exchange Protocol should specify the expectations for reliance on and use of WSDL.

The complexity and diversity of the Beta Phase service requests presented challenges for documenting and communicating the transport and exchange requirements. For the Network to work well, Partners must be able to communicate a common set of transport and exchange requirements for each Flow. As Partners manage more services, documentation and tools for

---

[12] See "Building on the Beta: a Proposal to the Network Steering Board for a Beta Follow-on Project."

[13] With a few exceptions, these interactions were between the Java test application and the Nodes because the Beta Phase did not include the development of requestor Nodes. However, the same issues applied for the few Node-to-Node interactions that took place, and will apply to future Node-to-Node exchanges.

information request and exchange management will become increasingly important. Again, Partners need the Node Functional Specification and Network Exchange Protocol.

Connecting to Existing Information Systems (High-level Overview and Lessons Learned)

Overview

The Beta Phase Nodes accessed data from the information systems that Partners already use to store and manage environmental data. These systems vary in complexity and architecture. Most Beta Phase participants spent a majority of their project time connecting their Nodes to their existing systems.

Lessons Learned

Data mapping between the specific service-request (as outlined in the Schemas) and the tables in the agency databases consumed the most time. It required careful implementation, a combination of technical and programmatic expertise, and close communication between the participants' Node teams and data administration staff. In some cases, participants' application integration tools worked for connecting Nodes to their existing information systems. Finally, Partners would benefit from recommendations on the most effective ways to query data and how to support flexible querying (e.g., ad hoc queries).

In general, most Partners will have mixed architectures of integrated, federated, warehoused, and/or isolated systems. In the context of Nodes, these differences will influence where and how Partners connect their information systems to their Nodes.

Beta Node Performance (High-level Overview and Lessons Learned)

Overview

The Beta Phase included two Node performance evaluations:

<u>Performance that measured the total transaction time.</u> (See Figure 4.) The total transaction time has five steps:

1. The requestor Node (or, in the case of the Beta Phase, the Java test application) sends a service request to the requested Node.
2. The requested Node processes the service request and sends the resulting query to the information system.
3. The information system processes the query and sends the data back to the requested Node.
4. The requested Node processes the data and sends the service response back to the requestor Node.
5. The requestor Node receives the service response.

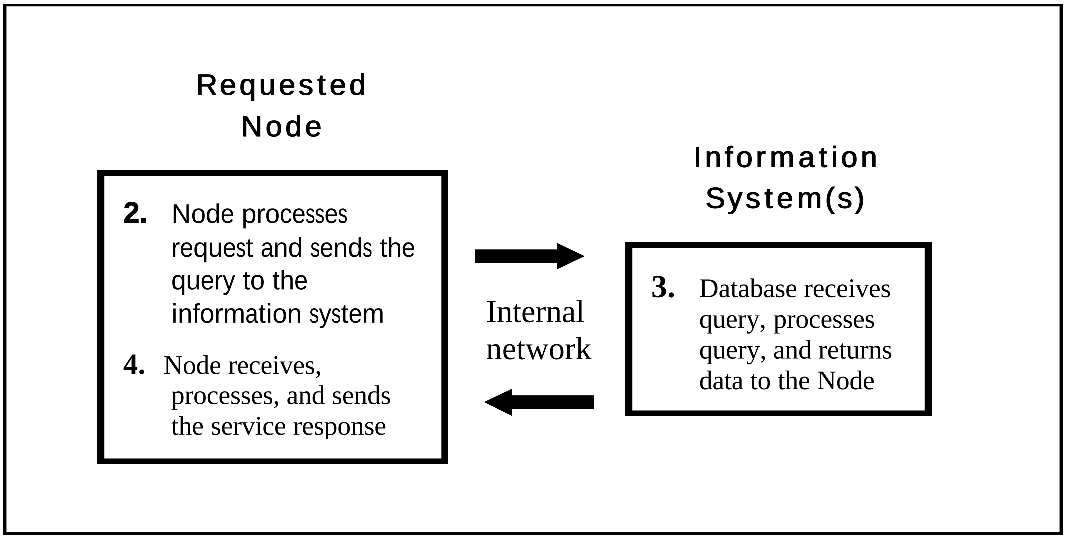


Figure 4. Steps In Node Processing Time

<u>Performance that measured the Node processing time.</u> (See Figure 5.) The Node processing time is a subset of the total transaction time (described above). The Node processing time *begins* when the requested Node receives the service request and *ends* when requested Node sends the service response.

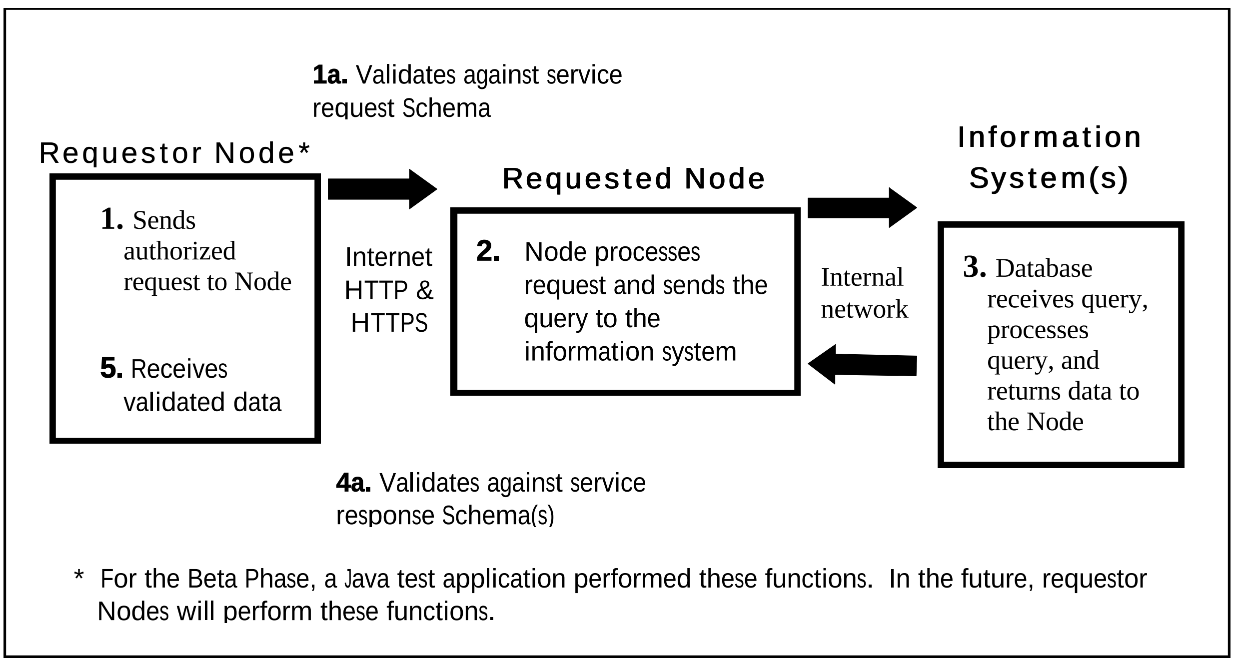


Figure 5. Steps Included in Total Beta Phase Transaction Time

Lessons Learned

From high-speed Internet connections, the total transaction times were acceptable: the smallest requests (for one specific record) took one second or less. Transactions of a few megabytes took between 30 seconds and a few minutes. Within the total response time, the Node processing time (i.e., not the Internet time) dominated the total transaction time. It is important to realize that while the speed of the Network is important, an implicit design assumption of the Network is that current batch transactions that produce large data sets could be accommodated by increasing the frequency of transactions thereby reducing both delays and file sizes. Obviously, the transaction requirements of some Flows may not be amenable to this approach. In those cases, other performance solutions may include compression or alternative transport protocols.

The secure Hyper Text Transfer Protocol Secure (HTTPS)[14] transaction times were slower by approximately 20 percent. (Note that these were exchanges between a Hyper Text Transfer Protocol (HTTP)[15] request and HTTPS response.) External (non-Node) factors that also affected performance included the following: Internet connection speed (this is likely to be the largest performance variable), network type, existing system congestion, existing system hardware, database structure and tuning, and query method. Finally, how Partners' choose to query their information systems can affect performance.

The following sections cover these technical topics in more detail.

F. Beta Node Functions

Understanding what Nodes *do* – the functions they perform – is as instructive as learning what hardware and software comprise Nodes. Like the Network, Nodes exist only to provide certain essential Network functions.

As shown in Figure 6, specific Node function "sets" fall into three categories: functions performed while requesting information (requestor Node functions), functions performed while fulfilling information requests (requested Node functions), and general Node management functions. The Beta Phase focused on requested Node functions, as these are likely to evolve first to allow Partners to send information to EPA (and generating requests using a test application was relatively easy). Future Nodes will also perform requestor functions.

Requested Functions

The Beta Nodes perform four major requested Node functions: 1) Responding to requests for information from other Nodes, 2) Receiving and processing requests by formulating queries to its attached existing information system(s), 3) Receiving and processing result sets from the information system(s)), and 4) Receiving and validating incoming requests, as well as processing any error message that may be sent from the information system(s) in answer to its queries. Figure 2 shows the functions performed by Nodes in the Beta Phase, as well as functions the future Nodes will perform. (Dotted functions are anticipated.)

---

[14] **HTTPS** – HTTP that provides for the secure exchange of information by using SSL as a sublayer.
[15] **HTTP** – HyperText Transfer Protocol is a protocol used to request and transmit files, especially webpages and webpage components, over the Internet or other computer network.

Once the query of the existing information system(s) is complete, the Node receives the query results. The Node processes the results to conform to the agreed-upon data format (XML, if it is not already in XML format), and validates the XML file against the pre-defined Schema(s). If the result is valid, the Node packages it in a SOAP (transport) envelope and applies the required level of security before transmitting it via the Internet to the requestor. (Note that, in the future, Schema validation may occur up to four times during this process: twice by the requested Node and twice by the requestor Node.)

The requested Node also handles erroneous requests. Validation against the service request Schema determines if the request is acceptable/valid. The Node verifies the incoming request, strips it of its SOAP transport envelope, processes the request, and validates the request against the service request Schema. If validation fails, the Node creates an error message, packages the message in a SOAP envelope, applies the required security, and transmits the response (error message) to the requestor.

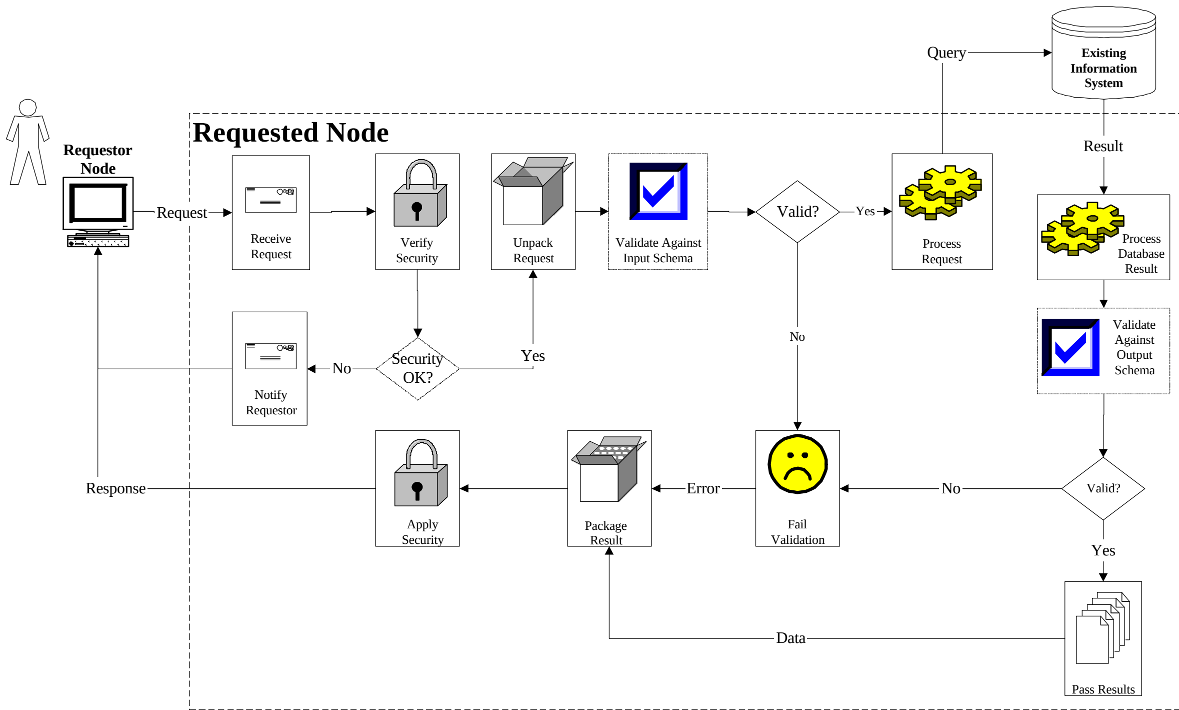


Figure 6. Requested Node Functions (Beta and Anticipated Future)

Some requests are valid, but result in errors when the information system(s) execute the queries. When this occurs, the requested Node receives an error message from the information system(s) rather than a result set. In the Beta Phase, these errors do not match the Schemas and therefore validation fails. When this occurs, the Node creates an error message, packages it in a SOAP envelope, applies the required security, and transmits the result (error message) to the requestor.

Beta Service Requests

As described in the "Introduction," the Beta (requestor) Nodes processed the three types of service requests outlined in Table 4. Participants called these requests, "get facility__" [*By Change Date*, *By Parameter* (i.e., either by facility ID or by Environmental Interest type), and *By ID]*. The Beta Nodes received these requests and responded with either the corresponding requested information or an error message.

Table 4. Beta Phase Service Requests

Request Name ("Get facility…")	Description	Schema(s)*
By Change Date	Returns facilities with any data changed as of the defined date	Component Facility Schemas
By Parameter	Returns facility records for facilities that match on Facility Name (*FacilitySiteName*) *and/or* Environmental Interest (*EnvironmentalInterestType*)	Abbreviated Facility Schema
By ID	Returns facility records for facilities that match on state facility ID (*StateFacilityIdentifier*)*or* federal ID (*FacilityRegistryIdentifier*), and StateUSPSCode	Consolidated Facility Schema

* "Appendix 7-Data Exchange Templates" for more detail on the Schemas used for the Beta Phase. (Note that, for the Beta Phase, a Java test application performed Schema validation.)

Requestor Functions

Future Nodes will perform three major requestor functions: 1) Generating requests for information from other Node, 2) Receiving and processing responses, and 3) Receiving and processing any error messages. Figure 7 represents the anticipated functions that will be performed by requestor Nodes. (For the Beta Phase, a Java application substituted for the requestor Node.)

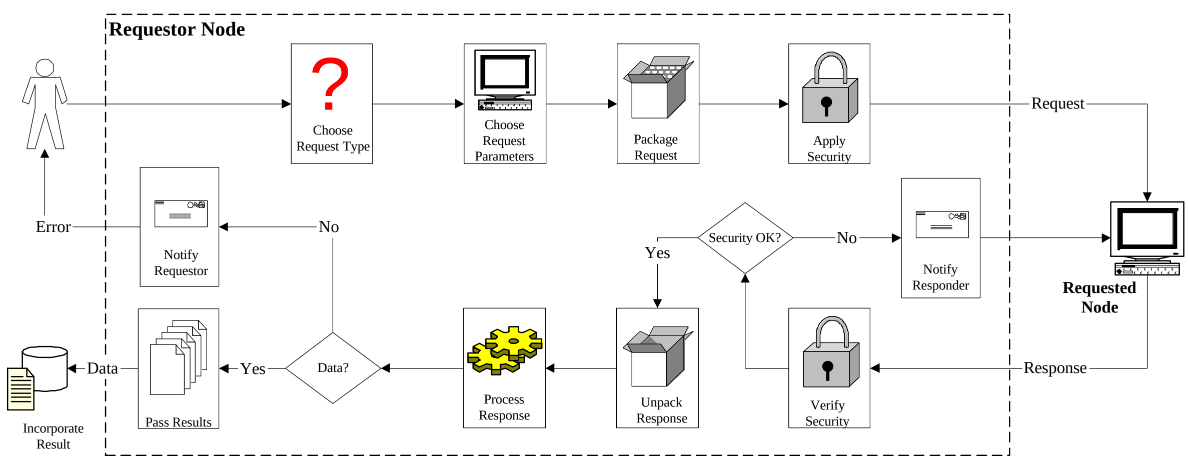


Figure 7. Anticipated Requestor Node Functions

Requestor Nodes may create requests in several ways. They are likely to send standard requests set to run on a specified schedule. Alternatively, an external application like the Java test application used in the Beta Phase, *could* allow Partners to send valid requests without relying on a requestor Node. This would support "on the fly" integration of data. Partners will generally use requestor Nodes to choose the type of request, select any limiting parameters, package the request for transport, add security to the request, and transmit the request via the Internet to the Node that requested information.

Once a request has been fulfilled by another Node, the requestor Node receives a response to its request. When the requestor Node receives the response, it verifies that the response includes the applicable security measures. It will then process the (valid) data, which could include additional quality control or needed transformations, and store the result set for further use. These uses may include data loading (into an existing information system), display, or distribution.

In some cases, the response that arrives will be an error message. This message indicates an error in the request or response, (i.e., the data set is not in a valid format). When this happens, the Node will still verify that the error message does not pose a security threat before unpacking the result from it transport envelope, processing the response, and responding as appropriate.

Lessons Learned and Issues

- The Beta Phase only covered certain Node functions. Additional work should be performed to investigate the full range of anticipated Node functions, including managing the exchange process and generating service requests.
- The place and timing of many functions are variable and can be rearranged depending on a Partner's needs. Further attention needs to be given to the timing and placement of security functions and validation.
- Overall, Node functions should be clearly defined to maximize consistency, while allowing flexibility in their implementation. To achieve this, Beta Phase participants strongly recommend the development of a Node Functional Specification.

G. Beta Node Architecture

The basic architecture for all Nodes is the same because all Nodes have to perform the same high-level functions. Each Node must have an Internet connection, be able to perform all of the functions discussed in "Beta Node Functions," and have a functional connection to an information system. Detailed Node specifications, however, differ depending on the specific Node hardware and software.

It is important to understand that the functions a Node performs and the way a Node is constructed do not have to have any prescribed physical relationship. While many Partners will construct their Node on a single web server, Node applications may also be distributed among more than one server, based on the hardware and network topology[16] of the Partner.

Nodes are composed of several components (see Figure 1). The Basic architectural components of a Node include the following:

[16] **Topology** – The specific physical arrangement of the elements of a given network. These elements include physical interconnections, distances between nodes, transmission rates, and/or signal types.

- Request Listener: requests and data are currently being packaged in a SOAP envelope for transport. The listener needs to actively access incoming requests to the Node.
- Request Processor: a tool that unpacks requests for further processing by the XML processor.
- XML Processor: The XML processor requests and data are formatted and defined in XML Schemas. The processor must be able to interpret XML requests, translate requests and data into prescribed XML Schemas, and validate requests and output data against defined XML Schemas.
- Middleware[17] for Data Mapping: middleware maps the data in the XML Schema to the data in the participant's existing information system. Not only does it identify the correct data elements, but it also translates data types and formats as needed.
- Database Connectivity: database connectivity tools allow software, like middleware, to connect to and communicate with the existing database.

These architectural components work together to provide full Node functionality.

Approaches to Beta Node Architecture

The Beta Phase participants approached Beta Node architecture with three general technical toolsets. The toolsets for each approach were anchored by one of the following: Microsoft BizTalk 2000, used by DNREC, NHDES, and UDEQ; Oracle 9i Application Server, used by FDEP; and XAware XA-iServer, used by NDEQ and NMED.

Beta Phase participants' varied the details of the implementation to fit their specific needs and computing environments. Additionally, Beta Phase participants varied the data sources for their Beta Node implementation based on their needs for data security, access issues with network firewalls, and structure of the query.

Summary of BizTalk Beta Node Implementation

Three participants established Nodes using BizTalk: DNREC, NHDES, and UDEQ. While the basic Node components were the same, each participant had unique existing information systems and different vendor-specific hardware and software. Table 5 summarizes the architecture of the Beta Phase Nodes.

[17] **Middleware** – a broad array of tools and data that help applications use networked resources and services.

Table 5. Summary of Architectural Components for Beta Phase Participants Using BizTalk

Participant	SOAP Listener	SOAP Processor	Object Handler	Data Mapping	XML Processor	Database Connectivity	Database
DNREC	IIS 5.0/ ISAPI Listener	MSXML 4.0	COM +	BizTalk 2000 w/ AIC component	BizTalk 2000	SQL ODBC	MS SQL Server 2000
NHDES	IIS 5.0/ ISAPI Listener	MSXML 4.0	COM +	BizTalk 2000 w/ AIC component	BizTalk 2000	Oracle 8i ODBC *	Oracle 8i
UDEQ	IIS 5.0/ ISAPI Listener	MSXML 4.0	COM +	BizTalk 2000 w/ AIC component	BizTalk 2000	Oracle 8i ODBC *	Oracle 8i

* A version 9i database driver might result in performance gains. It would prevent data from being returned in packages (required by the 8i driver and ADO).

BizTalk Architectures

Figure 8 shows the generic architecture implemented for BizTalk users. Note that the majority of the architecture is Microsoft-based, including Internet Information Server (IIS), MS XML, BizTalk, and ActiveX Data Objects (ADO)[18]. This Node architecture allows access to any database that can communicate through Open DataBase Connectivity (ODBC)[19].

[18] **ADO** – ActiveX Data Objects is database connectivity tool based on Microsoft's ActiveX technology.
[19] **ODBC** – Open Database Connectivity is a standard database access method developed by Microsoft Corporation which is interoperable with most database management systems.

Figure 8. Generic BizTalk Node Architecture

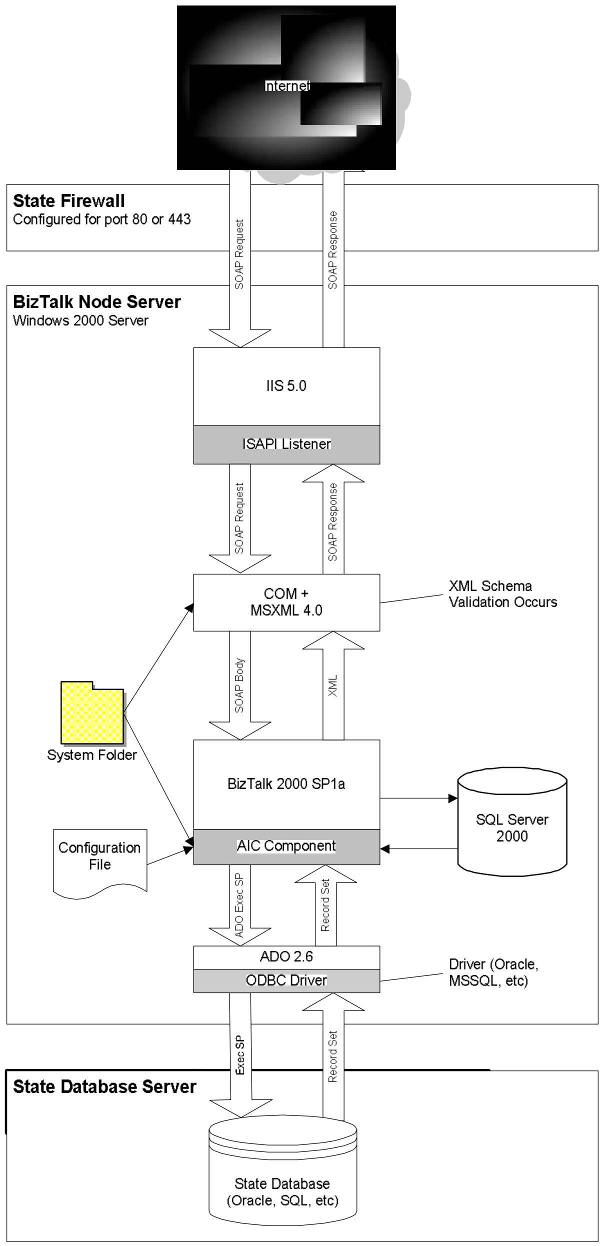


Table 6 summarizes the purchased and custom software required for the Beta Phase BizTalk Nodes to function. It also includes the configuration settings and the Node database connectivity components utilized

Table 6. BizTalk Node Software and Configuration

Shrink Wrap Software	Custom Software and Directories	Node Configuration Setting and Accessibility	Node Database Connectivity
Windows 2000 SP2 (w/ MSMQ 2.0, IIS 5.0)	[SOAPNode]<dir>	Terminal Services Access – To allow remote access, configuration and support	ADO
Microsoft SQL 2000 SP1	BizTalkSoapAIC.dll	File Transfer Capabilities to allow custom software to be uploaded to the machine.	SQL ODBC Driver
Microsoft BizTalk 2000 (evaluation, standard, or enterprise edition	BizTalkSoapEnabler.dll	URL established and port 80 and 443 open from the BizTalk Node to the internet.	Oracle 8i ODBC Driver
Microsoft Software Installer 2.0 (need for xml 4.0 kit)	BuildXMLOutput.dll	SSL certificate configured for port 443.	
Microsoft XML 4.0	NodeCfg.xml (DB connection info)	Network connectivity and database access from the BizTalk node to the state database.	
Microsoft SoapToolkit 2.0 SP2	temp_xml_out<subir> (all temporary XML files stored here)		
Microsoft MDAC 2.6 (include ADO 2.6)	BizTalk Ports and Channels		

Example of BizTalk Beta Node Implementation

DNREC's Node is configured as shown in Figure 9. All servers in the system use the Windows 2000 Advanced Server operating system, and the Node server uses BizTalk Standard Edition as the Node engine. The production database is located in DNREC's main office building on a data server and is on the State of DNREC's Intranet. Production data is replicated to a mirrored database, located in the State's de-militarized-zone (DMZ)[20] that resides between the Intranet-DMZ firewall and the DMZ-Internet firewall. Data is replicated using transactional replication over a 100 MIPS fiber line, with an average latency of less than seven seconds. From the DMZ database server, data is passed to the BizTalk server over a 100 MIPS connection. The BizTalk server is a Dell PowerEdge Dual Pentium, running at 1 GHz with 1 GB of memory and three 18 GB hard drives in a RAID 5 configuration (Figure 9). The DMZ servers are located in the State's computer center, about three miles from DNREC headquarters. The DMZ is managed by the

[20] "DMZ" refers to a zone "between" firewalls that is more exposed than internal systems, but still provides at least one layer of protection.

State computer center and is not under direct control of DNREC. Connection to the Internet from the State computer center is over a DS3 line. Information to fulfill the request is returned using Stored Procedures[21] that outputs record sets with field names matching the Schema being used for the Beta Phase.

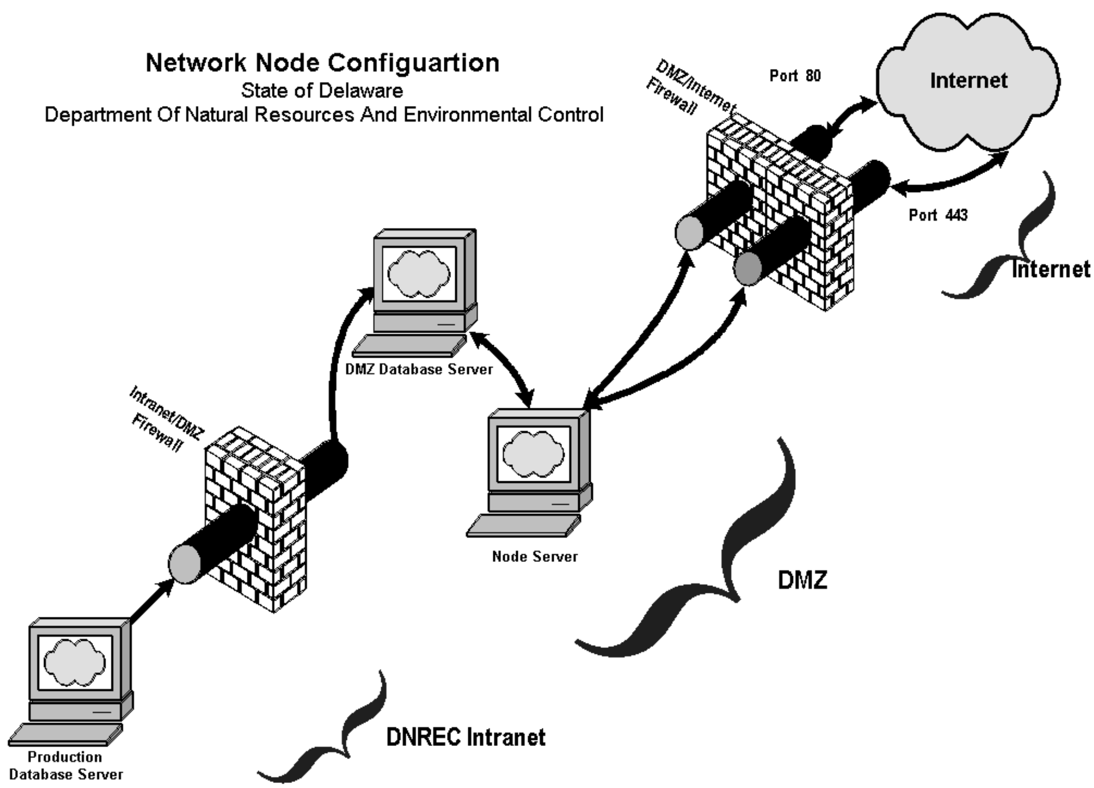


Figure 9. Delaware Natural Resource and Environmental Control Beta Node Architecture

Oracle Beta Node Architecture

Oracle 9i Application Server (9iAS) using Java provides Node functionality. Further, the Oracle toolset can take advantage of the native XML capability of Oracle's 9i Database Server by handling and formatting data in XML.

[21] **Stored Procedures** –sets of SQL statements with assigned names that are stored in databases in compiled form so that they can be shared by a number of programs.

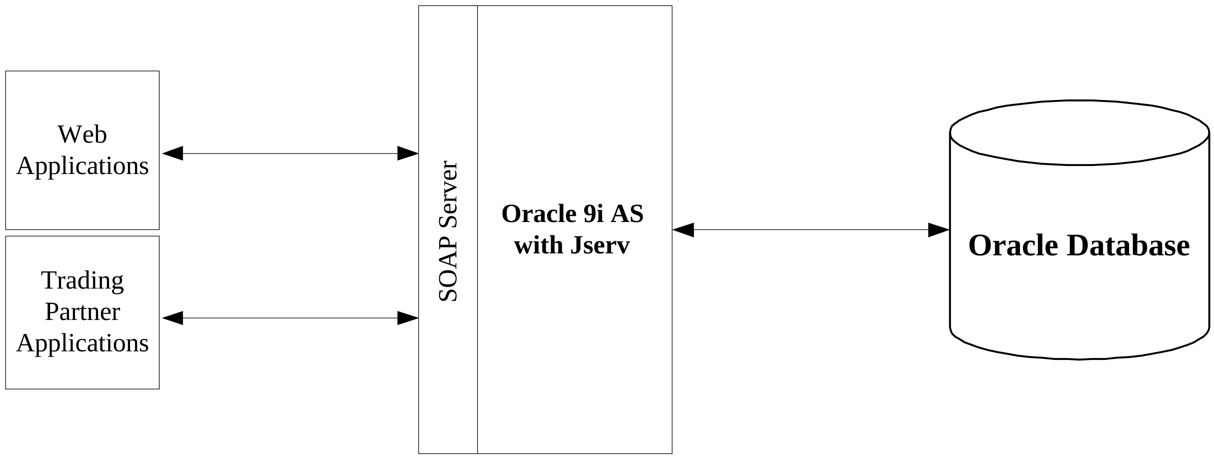


Figure 10. Generic Oracle Node Architecture

Example of Oracle Beta Node Implementation

FDEP developed a Node using JAVA, a SOAP server, Oracle 9iAS with Apache Jserv, and Oracle Database.

The client application sends an XML request in a SOAP envelope to the Node over the Internet. The firewall received the request and diverted it to the Oracle 9iAS Apache and JServ SOAP Server. The Node unpacked the request and sent the query to the Oracle Database. The database sent the results, formatted in XML, to the Node, and the Node sent a response (results set) back to the client through the firewall.

The Node server architecture consists of three components:

- Web server with firewall: receives the request over the Internet and redirects it to Oracle 9iAS after the security validation.
- Oracle 9iAS with Apache JServ & SOAP Server: receives the request through the firewall, retrieves the data, and responds to the client after processing the request.
- Oracle Database 8i/9i: receives the query from the servlet through the internal firewall and returns the corresponding data.

The Node middleware consists of Oracle 9iAS, in which the Apache JServ, SOAP server, and related JAR files are enclosed. Oracle 9iAS accesses the web server, and within its context, the Apache JServ runs the servlet. The SOAP server reads and unpacks the SOAP envelope and creates the SOAP envelope for the response. The required JAR files and CLASSPATH settings were modified to integrate Oracle 9iAS, Apache JServ, and SOAP server to synchronize.

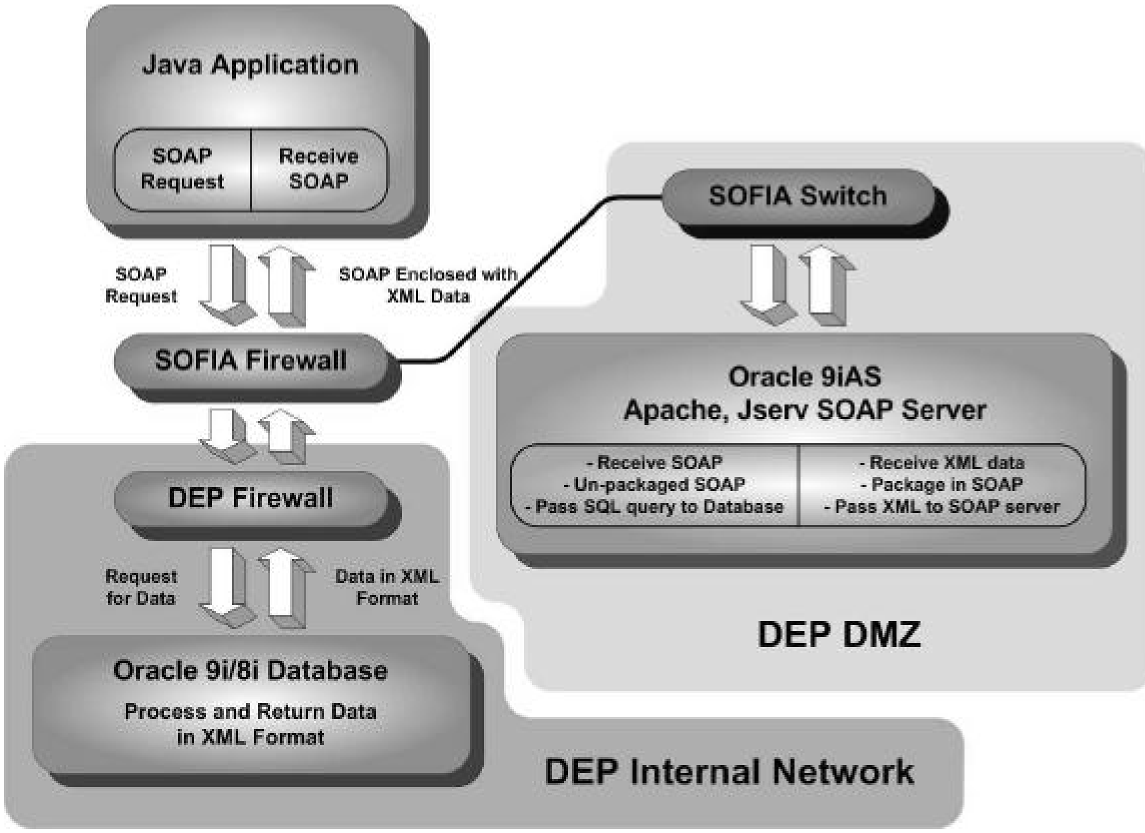


Figure 11. FDEP's Beta Node Architecture

Figure 11 shows the architecture and topology for the FDEP Beta Node implementation. For the Beta Phase, requests were sent and responses received by a Java test application. Table 7 contains a summary of FDEP's Node component.

Table 7. Summary of Architectural Components for Beta Phase Participants Using Oracle

Participant	SOAP Listener	SOAP Processor	Object Handler	Data Mapping	XML Processor	Database Connectivity	Database
FDEP*	Jserv Server/ servlet	SOAP Server/ Jserv servlet	Oracle 9iAS	Oracle 9iAS/ Apache Jserv	Oracle 9i AS	Oracle 9i Native Connection	Oracle 9i

*FDEP had multiple Node installations, each with slightly different components.

XAware Beta Node Architecture

XAware includes an application server (XA-iServer) and toolset designed to support the exchange of data using XML.

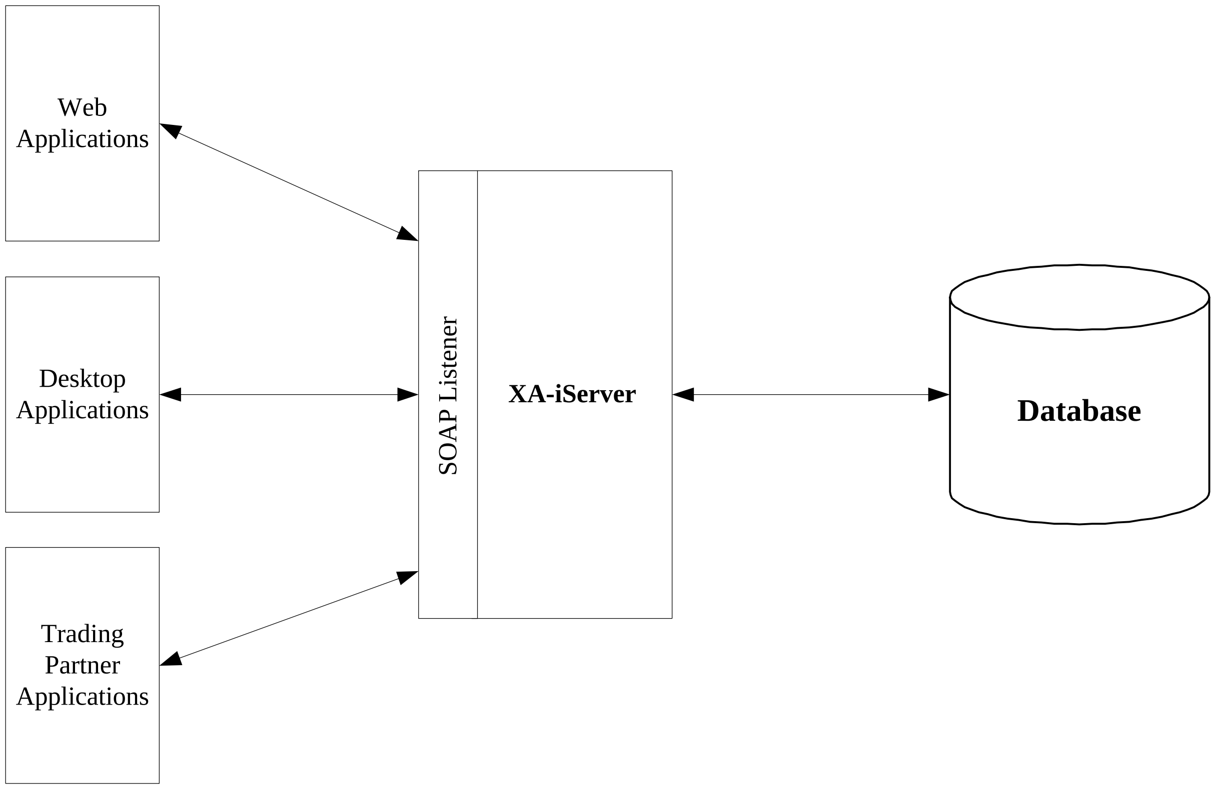


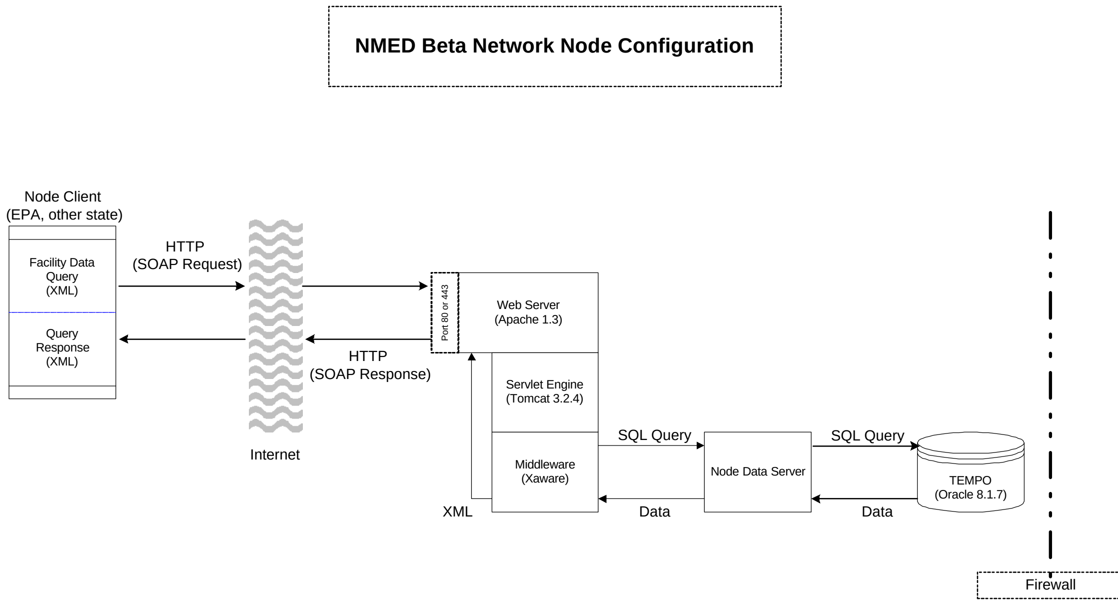
Figure 12. Generic Architecture for XAware Implementations

Example Beta Node Implementation

The NMED Beta Node is based on middleware from XAware software running on an Apache web server. XAware's XA-iServer processes an inbound SOAP envelope containing an XML formatted client request. The inbound request is validated against an input Schema file to ensure that all required elements are included and properly formatted. XA-iServer then generates a SQL[22] query based on the parameters supplied by the requesting client and accesses the TEMPO Oracle 8i database. Data returned to the XA-iServer is formatted into an XML document, validated against an output schema, wrapped in a SOAP envelope and returned to the requesting client. Please refer to Figure 13 for a High-Level diagram of NMED's Node architecture.

[22] **SQL** – Structured Query Language is the standard language for relational database management systems.

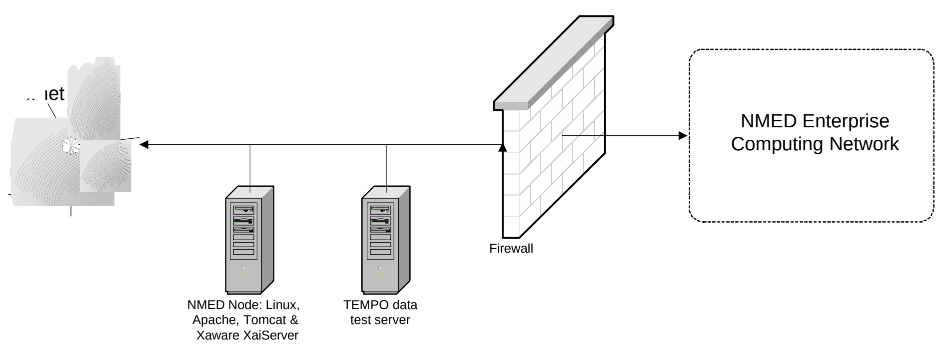
Figure 13. High-Level Node Architecture for NMED



NMED has a network infrastructure that includes both local and wide-area networks (LAN and WAN) running on UNIX and Windows NT servers. Workstations are primarily Windows based and run operating systems ranging from Windows 98 through Windows XP. World Wide Web and FTP servers are located in an unsecured area of the network (the "DMZ"), which is accessible to computers external to NMED. NMED maintains a public web site (www.nmenv.state.nm.us) that provides static content to the public and regulated entities. NMED's World Wide Web capabilities are currently built upon Microsoft IIS running on a dual processor Pentium class server. The agency is developing a web portal using Java 2 Enterprise Edition technologies running on Sun Solaris servers. The NMED Intranet server resides behind the firewall. No HTTP access is allowed from the outside world to machines inside the NMED firewall.

The NMED Beta Node is configured on the Department's LAN as illustrated below in Figure 14:

Figure 14. NMED High-Level Network Typology



For the Beta Phase, NMED utilized stand-alone web and database servers located in the DMZ. The NMED node is configured on an Intel-based personal computer with a 200 MHz Pentium I processor, 160MB of RAM and 20GB of hard disk space. The operating system is Red Hat Linux distribution 7.1. The web server and servlet engine are Apache web server version 1.3 and Tomcat servlet engine version 3.2.4. The XML middleware is XAware XA-iServer. The NMED Node is isolated from other NMED servers with the exception of the public FTP, web, and the Node data servers. The Node data server was installed in the DMZ to enable access to data while reducing the risk of intrusions. The Node data server is a dual processor Intel-based computer running a test instance of the TEMPO Oracle 8i database.

Summary of XAware Beta Node Implementations

Variations in the XAware implementations were primarily database and database connectivity. The following table summarizes the components used by Beta Phase participants that used XAware.

Table 8. Summary of Architectural Components for Beta Phase Participants Using XAware.

Participant	SOAP Listener	SOAP Processor	Object Handler	Data Mapping	XML Processor	Database Connectivity	Database
NDEQ	Tomcat	XAware	XAware	XAware	XAware	JDBC	DB2
NMED	Tomcat	XAware	XAware	XAware	XAware	Oracle ODBC	Oracle 8i

Lessons Learned

> **Corresponding Network Implementation Plan Milestone:**
>
> In 2002, the Board will develop three Technology Templates (Demonstrated Node Configurations) for State Nodes
>
> (*"Guiding Network Implementation and*)

- Each Node can be different, even though they perform the same functions.
- The physical architecture of a Node is not dictated by its functions; rather, it is influenced and determined by the architectural needs of the Partner. This means that Node functions can be distributed among several software packages and hardware platforms, or they could reside in a single product suite and server.
- Partners should make every attempt to fully utilize the tool that they have chosen. For example, very little of the BizTalk functionality was used during the Beta Phase. Leveraging the strengths of the selected tools will reduce custom work and ensure consistent Node behavior.

- Using existing State hardware and software allows small State IT groups to leverage existing technical expertise and physical resources to create a Node. Analysis and projections indicate that a fully functional Node will not require more than 25 percent of the existing State Biztalk server's capacity in UDEQ.

Corresponding Network Implementation Plan Milestone:

In 2002, the Board will assign responsibility for developing and disseminating technology trends information.

(*"Monitoring and developing recommendations on Network Technologies"*)

Beta Node Network Topology

The Beta Phase participants implemented their Nodes within a number of topologies. While the basic structure and components were the same, each arranged hardware and software components in different logical layers on their network, depending on existing infrastructure, data and server security requirements, and component capabilities. In general, each participant created or worked within an arrangement of firewall(s), application server(s), and data server(s) to create their Nodes and relate it to their networks.

Three Examples of Node and Network Topology

UDEQ physically separated the Node components, allowing the SOAP listener and BizTalk to reside on a State of UDEQ server dedicated to BizTalk, while data mapping occurred on a separate application server at UDEQ. Both the State and UDEQ servers were located behind the State DMZ. See Figure 15

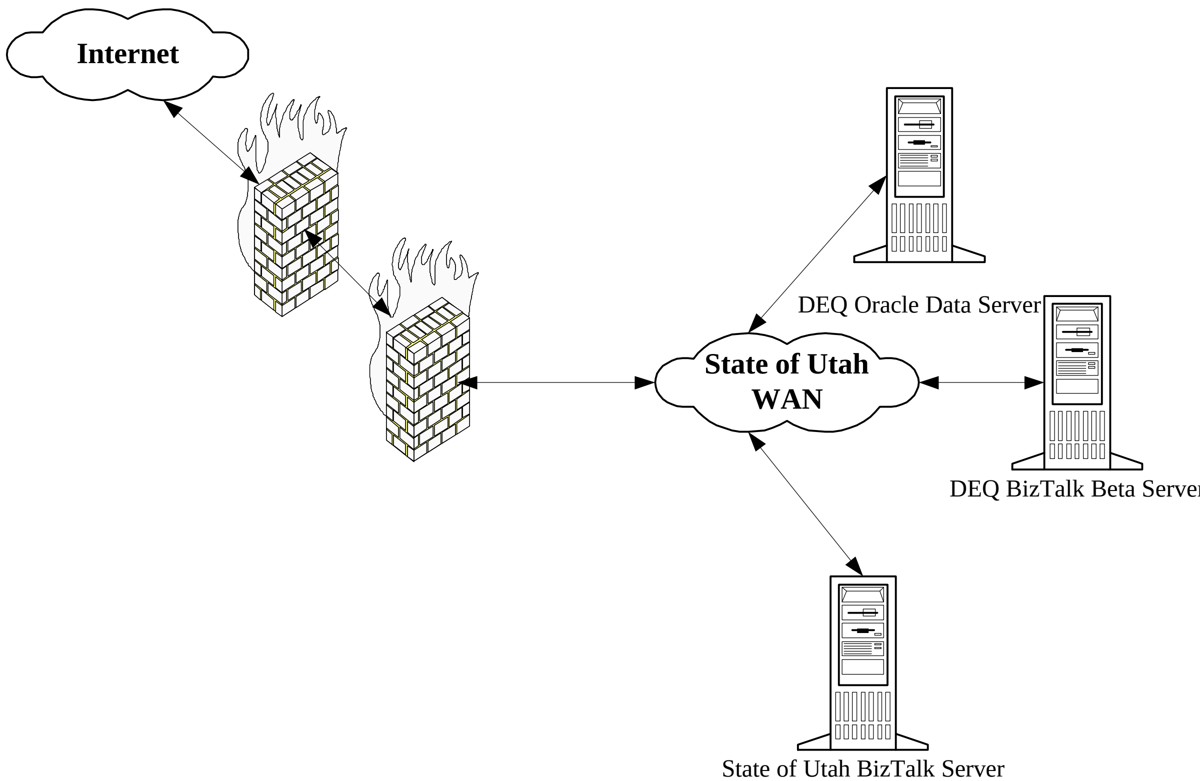


Figure 15. Utah Node and Network Topology

While this arrangement posed no serious technical challenges, UDEQ's experience showed the need for heightened coordination between UDEQ staff, network staff, and State computing staff. Inter-agency coordination requirements made managing priorities and keeping on schedule more difficult.

FDEP's Node topology placed the application server in the DMZ, and the data server on the DEP network. This arrangement required several trips through various firewalls for each request and response. An incoming request must be received by the Internet firewall and routed to the Oracle 9i Application Server (AS) running Apache Jserv. The application server then processed the request, formulated an SQL query, and passed the query through the Intranet firewall to the Oracle database. The database then retrieved the requested data, formatted it in XML, passed the file out through the Intranet firewall to the application server, packaged in SOAP, and transmitted through the Internet firewall to the requestor. (See Figure 11.)

NDEQ's Node and data source are both located on the DEQ network. Access to the Network is through the State of NDEQ backbone. The backbone is protected by a firewall, and the IBM servers at DEQ provide an additional layer of security. Requests sent to the NDEQ Beta Node are received through the connection with the State backbone, and are processed by the Node server through either Port 80.

Summary of Topology

Table 9 summarizes the variation in Nodes and Network topologies for the Beta Phase participants. The variation shows that Nodes can be implemented in many different network environments and still perform their required functions. Because of this variation Node implementers will need to coordinate with several groups.

Table 9. Key Components of Beta Phase Node Topology

Participant	DMZ	SOAP Listener Location	Middleware Location	Data Source Location	Node Data Source
DNREC	Y	Application Server in DMZ	Application Server in DMZ	Data Server in DMZ	Replicated Production Database
FDEP	Y	SOFIA Firewall	Application Server in DMZ	DEP Network behind DMZ	Summary Database from production systems
NDEQ	N	Application Server behind firewall	Application Server behind firewall	Data Server behind firewall	Production Database
NHDES	N	Application Server behind firewall	Application Server behind firewall	Data Servers behind firewall	Production Database
NMED	Y	Application Server in DMZ	Application Server in DMZ	Data Server in DMZ	Replicated Production Database
UDEQ	Y	State Application Server behind DMZ	DEQ Application Server behind DMZ	DEQ Data Server behind DMZ	Summary Data Warehouse from production systems

Topology Lessons Learned

- It is feasible to use physically separated components. In UDEQ, the BizTalk server is managed by centralized State information technology services. There can be, however, administrative and coordination issues created by this type of configuration. Principle coordination issues are security and problem resolution.

EPA (CDX) Node Testing and Prototype Architecture

Corresponding Network Implementation Plan Milestone:

In 2002, EPA will prototype the EPA Out-Node Operational.

(*"Establishing Network Nodes"*)

Like their State counterparts, CDX staff used the Beta Phase to test and evaluate XML tools. As of the writing of this report, CDX is still determining the architecture of its prototype Node. CDX intends to complete implementation of its first generation Node by the end of this year. A prototype of CDX's Node will be functional in the near future.

> **Corresponding Network Implementation Plan Milestone**:
>
> CDX is expected to receive Flows into its National Systems by the end of 2004
>
> (*"Establishing Network Nodes")*

At the moment the CDX Node does not use a commercial middleware product. However, for testing purposes, CDX developed a Node using Microsoft XML/SOAP toolsets that interface with an Active Server Page (ASP)[23] hosted on a Microsoft IIS. The ASP intercepts the inbound service request and hands it off to a Component Object Model (COM)[24] component for identification of the Node's service request type, and initiation of the appropriate query to an Oracle 9i database. The COM component also converts the Oracle data to XML and prepares a SOAP response. Table 10 contains a summary of the CDX Node architectural components.

Table 10. Summary of CDX Node Architectural Components

Participant	SOAP Listener	SOAP Processor	Object Handler	Data Mapping	XML Processor	Database Connectivity	Database
EPA CDX	Custom ASP	Custom COM object/ MS SOAP toolkit	Custom Com Object/ MSXML 3.0	Custom Com Object/ MSXML 3.0	Custom Com Object/ MSXML 3.0	Oracle OLEDB Oracle 9i Native Connection	Oracle 9i

The Node Architecture consists of:

- Web Server with Firewall: CDX has a redundant T-1 Internet connection, a firewall with integrated virus detection, and network intrusion software for protection. Each server also has virus detection software installed. The firewall will allow "wh standard HTTP requests and an encrypted exchange via Secure Sockets Layer (SSL)[25] using HTTPS.

- Node Server: the CDX Node web server is a Microsoft IIS 4.0 on a Compaq Proliant DL380R01 box. As stated previously, the CDX Node does not use a commercial middleware product. A custom COM object interprets the inbound request and evokes the appropriate Oracle stored procedure with the parameters provided in the inbound SOAP request. The COM component also converts the Oracle dataset into the XML data before wrapping the XML data in a SOAP envelope for the return trip to the requestor. The COM component interacts with the Microsoft XML Toolkit and the Microsoft MSXML

---

[23] **ASP –** Active Server Page is a server-side scripting technology that can be used to create dynamic and interactive Web applications.
[24] **COM** – The Component Object Model Component Object Model is Microsoft's framework for developing and supporting program component objects. COM provides the underlying services of interface negotiation, life cycle management, licensing, and event services.
[25] **SSL** – Secure Sockets Layer created by Netscape Communications, also known as secure server, provides for the encrypted transmission of data across the Internet. Users on both sides are able to authenticate data and ensure message integrity.

3.0 parser. CDX installed both of these software packages on the CDX web server. CDX also installed the ASP page that accepts the inbound SOAP request from the Java test application on the web server.

- Oracle 9i database: this database houses a static subset of the EPA's FRS. The database contained over five million records in five tables, and as of this writing, was two Gigabytes in size.

Corresponding Network Implementation Plan Milestone:

The Board will establish responsibility and schedule for Network security guidelines development in 2002.

(*"Ensuring Network Security"*)

Beta Node Security

The security infrastructure for the Network will be based on the small set of core technologies, such as SSL and HTTPS. The Network seeks to have a common model for security that can be implemented by all Partners. In 2002, the Board will be developing specific guidelines for Network security standards. For more information on Network security please refer to the *Network Implementation Plan.* The Beta Phase tested levels 1 and 2 of Network's four proposed security levels. Eventually, Partners will use four levels of information security for Flows. Table 11 provides descriptions of the Network security levels and the Beta Phase use of them.

Table 11. Beta Phase Security Summary

Level	Security Level Description	Pilot Project Usage
1	Public information that requires no authentication or certification of integrity will be available through the Internet on a public, non-secure website.	The Alpha and Beta Phases used this level of security. Because these phases transported SOAP messages participant selected HTTP (Port 80) as the transport. FDEP and NDEQ used this level of security for the Beta Phase.
2	Information that requires some additional level of authentication (i.e., that it is the State environmental agency that is submitting the data) and a higher level of integrity protection will be available through the Internet on a website that is secured using SSL.	The Beta Phases used this level of security, specifically DNREC, FDEP, NHDES, and UDEQ. Because this level of security suggests using SSL, participant selected HTTPS (Port 443) as the transport. Participants accessed Port 443 via the use of server certificates, further described below.
3	Information at this level requires bi-directional authentication and a higher level of confidentiality protected by SSL at the server level, and requires users' digital certificates.	To date, the Pilot Project has not tested this level of security.
4	Information protection that requires non-repudiation, in addition to privacy, authentication, and data integrity, will be protected by requiring a digital signature "affixed" to the data, that can be validated at the time of acceptance of the information by the environmental agency or the external user.	To date, the Pilot Project has not tested this level of security.)

Beta Phase participant provided two URLs to access each Node: one URL is for HTTP that accesses Port 80, which does not require an increased level of security; the other URL is for

HTTPS that accesses Port 443, and requires an increased level of security (SSL). The HTTPS URL requires the use of security levels two or three and Server certificate. EPA provided server certificates to Beta Phase participants to access Port 443.

> **Corresponding Network Implementation Plan Milestone:**
>
> In 2003, the Board will commission an independent security assessment and the Network security protocols.
>
> (*"Ensuring Network Security"*)

Server Certificates[26], security components used in the Beta Phase, will likely be used on the Network. These act as agents that can be trusted by the involved parties to issue public key certificates and can guarantee the accuracy of the information contained in the certificates that they issue. In essence, server certificate will allow Nodes to identify authorized Partners and defined exchanges. They are required when using security Levels 2 and 3 that use SSL.

As shown in the Table 11, NDEQ and FDEP used Port 80 and security level one to exchange data with each other. DNREC, NHDES, NMED, and UDEQ used Port 80 and security level two to exchange data with NDEQ and FDEP.

Security Lessons Learned and Issues

- Security for XML is evolving and may influence future Network security protocols.
- Beta Phase participants learned that setting up additional levels of security not only involved additional technical steps, like installing server certificates, but involved cooperation from other parts of their organizations which required additional effort and coordination.
- The Beta Phase did not test Level 3. Beta Phase participants did not realize that Java test application would need to be programmed to test this security level. Testing Level 3 was a stretch objective for the Beta Phase, and participants decided not to pursue this objective.
- Server certificates are complicated, i.e., server certificates are not "user friendly." Beta Phase participants spent a fare amount of time simply applying for server certificates. In some cases, participants' applications did not go through, which required the participant to reapply.
- Coordinating network staff with node staff turned out to be the number one security issue experienced by Beta Phase participants. For instance, getting a connection through the firewall proved to be troublesome, network staff were hesitant to do this.

H. Beta Node Data Transport and Exchange

The Beta Phase participants implemented two transport/exchange mechanisms (XML and SOAP) and experimented with a third (WSDL). Because expectations for specific Node data transport and exchange processes were not pre-defined, participants' SOAP implementation was directed

---

[26] **Server Certificates** – digital certificates for the server used by browsers to authenticate websites.

by the kind of hardware and software used. Node-to-Node interoperability was affected by the different SOAP implementations and several "tweaks" had to be made to enable seamless exchanges.[27] The Network's ability to provide services to multiple Partners depends upon the interoperability of the products used to implement Nodes.

Implementing SOAP

The Alpha and Beta Phases used SOAP to transport requests and responses. However, as participants discovered during the Beta Phase, there are no clear conventions on exactly how the SOAP Specification should be implemented.

SOAP is an XML-based protocol for accessing services, objects, and servers in a platform-independent manner. SOAP is an electronic envelope consisting primarily of a header and a body (see Table 12). The body of information within the SOAP envelope carries the message. HTTP is commonly used for SOAP transport and is the only current normative binding, but SOAP is not restricted to use only with HTTP. In other words, SOAP is a messaging exchange model that can bind to any transport mechanism. For the Network, the transport mechanism is HTTP/HTTPS.

Beta Phase SOAP messages

Table 12 contains an example of a SOAP message generated by the Java test application and sent to a Node. In the future, Nodes will send such requests, though options like the Java test application could still be used.

Table 12. Service Request 2 in a SOAP Envelope

```
POST /xaware/servlet/XAEpaSoapServlet  HTTP/1.1
Content-Type: text/xml
Content-Length: 594
SOAPAction: ""
User-Agent: Java1.3.0
Host: localhost:8070
Accept: text/html, image/gif, image/jpeg, *; q=.2, */*; q=.2
Connection: keep-alive

<soap-env:Envelope xmlns:soap-
env="http://schemas.xmlsoap.org/soap/envelope/"
                   xmlns:xsi="http://www.w3.org/1999/XMLSchema-
instance"
                   xmlns:xsd="http://www.w3.org/1999/XMLSchema"
```

[27] With a few exceptions, these interactions occurred between the Java test application and the Nodes, because the Beta Phase did not include the development of requestor Nodes. However, the same issues did apply for the few Node-to-Node interactions that took place, and will apply to future Node-to-Node exchanges.

```
15 xmlns:xs="http://EPACDX.LMI.ORG/NodesPilotBeta/Schema">
16 <soap-
17 env:Header><xs:zippedAttachments>NO</xs:zippedAttachments></soap-
18 env:Header>
19
20 <soap-env:Body>
21 <xs:NodesServiceRequest soap-
22 env:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
23 <GetFacilityBy Change Date>
24 <AsOfChangeDate xsi:type="xs:date">2001/03/05</AsOfChangeDate>
25 </GetFacilityBy Change Date>
26 </xs:NodesServiceRequest>
27 </soap-env:Body>
28 </soap-env:Envelope>
```

The SOAP message has the following parts:

- A Transport Binding Definition (lines 1-8) that defines the transport of the message. The SOAP action header (line 4) is the command that indicates the intent of the SOAP message. If this line is blank, it means that the intent of the message can be inferred from the target of the POST[28] (line 1): the URI.
- An electronic envelope (lines 10-28) that describes what is in the message and how to process it. It is useful to think of the Transport Binding Definition as the "how," and everything that is contained in the envelope as the 'why'. Within the SOAP Envelope are the SOAP body and SOAP header.
- A SOAP body (lines 20-27) that contains the payload of the message. This instance initiates a query for all records with a change date of 03/01/2001 (lines 21-25). The body of a SOAP message can carry data or can carry a command such as a Remote Procedure Call (RPC)[29].
- A SOAP header (lines 16-18) that contains transactions and object references that are vital to the message, but not part of the payload. This instance tells the Node that the returned information, which will appear in the body of another SOAP message, should be native xml (not zipped) attachments (line 17).

To implement SOAP in the Beta Phase, participants used various tools that were either built into the middleware or installed separately. Without having a pre-established Network SOAP usage protocol, each middleware/vendor implemented SOAP a little differently, which resulted in interoperability problems. Beta Phase participants implemented SOAP using three different approaches to attach the payload (response data) to the SOAP body. The first approach added one child element to the SOAP body element, and the value of this one element was the entire response, URL encoded, as a string. The second approach, instead of having one child element in the body, the XML data was placed directly in the SOAP body. A third method was used for the

[28] **POST** – HTTP method that creates new object links to specified objects.

[29] **RPC** – Remote Procedure Call is a type of protocol that allows a program on one computer to execute a program on a server computer.

By Change Date type of service request, where the SOAP response was constructed using Multipurpose Internet Mail Extensions (MIME)[30] types as attachments to the response. The SOAP Specification allows for all of these approaches, but implementing all of them caused problems with interoperability. (see "Using WSDL to Specify and Communicate Data Transport and Exchange" for further discussion).

SOAP Lessons Learned

- SOAP is supported by available software and can function as a basic transport/exchange standard on the Network.
- Appropriate Network usage of the SOAP Specification must be defined to ensure consistent Node implementation and interoperability. The SOAP standard is still evolving, and even the simple requirements of the Beta Phase demonstrated areas where further Network specifications on the *use* of SOAP in the Network are needed. These areas include:
    - Use of the Soap header to carry Network processing instruction
    - Evaluation of the ebXML[31] extensions to SOAP that build-in many of the features that were manually added by the Beta Phase team
    - Conventions for exactly which options for attaching the payload to the message are to be used for that Flow type
- As with any new standard, Partners should be skeptical about "compatibility" claims of vendors. Although this situation will likely improve as products mature, only those SOAP features demonstrated by existing Partners as fully implemented should be adopted into the Network Exchange Protocol. In the Beta Phase, SOAP attachments did not match with the available version of Microsoft's SOAP Toolkit's capabilities, and required a manual work-around. Stability in next SOAP versions and further definition of specifications should reduce the number of interoperability problems.
- Because SOAP is supported by available software and can function as a basic transport/exchange standard on the Network, the question is less about why to include it in the Specification and Protocol, and more why not to include it.
- The Node Functional Specification and Network Exchange Protocol should specify the expectations for reliance on and use of WSDL.
- The receiving Node needs to know what approach to expect to be able to properly handle the message.

---

[30] **MIME** – Multipurpose Internet Mail Extensions is an extension of the original Internet e-mail protocol (Simple Mail Transport Protocol (SMTP)) so that the Internet client and server can recognize and handle data other than ASCII text.

[31] **ebXML** – Electronic Business XML initiative is a modular suite of specifications that enables enterprises of any size and in any geographic location to conduct business over the Internet.

Using WSDL to Specify and Communicate Data Transport and Exchange

As discussed above, one of the key problems faced by the Beta Node participants was describing exactly what incoming SOAP messages and responses a Node should expect. Although some of this was documented in project plans, Beta participants inferred much of the actual request/response format by trapping the actual output of the Java test application. One participant used WSDL, another emerging XML Specification, to help solve this problem. WSDL describes the format of a SOAP message that, in the Beta Phase case, carried the service requests and responses. As Network usage conventions for SOAP are defined, they can be described using WSDL, which is readable to both humans and machines.

The Beta Phase did not include a plan for using WSDL because doing so would have introduced yet another new technology (that organizers saw as a “bonus” rather than a necessity) into the mix. A major lesson learned is that for efficient and successful Network transactions, explicit documentation of Network Exchange Protocol is needed (e.g., how to handle service requests) in order for multiple implementers to achieve interoperability. The Network Exchange Protocol will include additional conventions on how to implement SOAP and WSDL (beyond the Specifications themselves).

FDEP created a WSDL file to configure a SOAP server to handle a Beta Phase service request. The EPA CDX Node built a query based on the Florida WSDL and was able to retrieve data successfully. Table 13 describes the process used by CDX to take a WSDL file and build a simple web page that requests data from a Node. Non-technical readers may wish to skip the details, but the key point demonstrated is that the two participants used WSDL to have their software write the code needed to implement a given service request. WSDL can provide the exchange protocol and in-code generation for services that invoke Nodes in human readable form. In the example in Table 13, CDX used WSDL to generate code automatically.

Table 13. Using WSDL to Create a Web Page that Pulled Data from a Node

CDX is developing an Oracle-based Node using the Oracle JDeveloper9i (JDeveloper) to construct web services based on WSDL files. This Node will use Business Components for Java (BC4J) to interact with the Oracle 9i database and includes a Java Server Page (JSP) and two Java classes to pass information successfully.

To create a web service from the FDEP WSDL, EPA CDX created:

1. A web service stub, in JDeveloper, using the Web Services Stub/Skeleton wizard. The wizard prompted the user for the URL of the WSDL. (In this case, the Florida WSDL. For the Network, it is likely that the WSDLs will be on the Registry/Repository). They accepted the defaults for the service to use and the code to create, and finished the wizard. No custom code was written.

2. A JSP using the JSP wizard in JDeveloper.

3. A webpage (using the JSP directives) that had a form and a text area for data. A form was created on the web page that allows the user to enter a date (format DD/MM/YYYY):

```
<form action="http://localhost:8988/EPA-FloridaNode-context-
root/florida.jsp" method="post"> Date: <input type=text
name="searchDate"> 
<input type="image" src="images/btn_search.gif" value="Search"
border="0" width="40" height="16" alt="Search Button"><br> </form>
```

Immediately following the form is a <TEXTAREA> that shows in the page only if the user has entered a value into the search field and submitted the page. This snippet of code uses the web services stub made in step 1 (above) to call the Florida Node's web service. The resulting data is printed in the TEXTAREA field through the code below:

```
 out.println( stub.getFacilityBy Change Date( d ) );
<% if ( request.getParameter("searchDate")!=null ) { %>
<br>
<textarea name="xml_data" cols="100" rows="15">
<%
florida.NodeServiceRequestStub stub = new
florida.NodeServiceRequestStub();
Date d;
DateFormat fmt = DateFormat.getDateInstance(DateFormat.SHORT,
Locale.US);
try {
d = fmt.parse( request.getParameter("searchDate") );
out.println( stub.getFacilityBy Change Date( d ) );
} catch( Exception e ) {
e.printStackTrace();
}
%>
</textarea>
<% } %>
```

4. A test of the web service by invoking the JSP from JDeveloper. This spawns an OC4J server in the tool that processes the JSP. In order to run this on a production web server, you simply need to deploy it to a web server that supports JSP's (OC4J, Tomcat, Oracle iAS w/JServ, Oracle iAS w/OC4J, etc.).

Lessons Learned

- WSDL files could be useful during Node implementations to document and specify a service request and response interface. Defining the interface in a standard way will prevent interoperability issues that occurred during the Beta Phase. This information should be included in the Network Exchange Protocols. The Follow-on Project should fully implement WSDL to validate this solution.
- Additional conventions are needed about how to implement SOAP beyond the SOAP Specification itself. WSDL can be used to document transport and exchange requirements quickly and efficiently, configure a Node for a Flow, and create applications that retrieve data from Nodes. As in the case with SOAP, further clarifications on the use of WSDL in the Network are needed.
- The Beta Phase used a very small universe of information requests, but even these presented challenges in documenting and communicating the transport and exchange requirements to Beta Phase participants. For the Network to grow, Partners must be able to communicate the transport and exchange requirements for each Flow as efficiently as possible. In addition, as Partners are managing more services, the documentation and tools for management of these services will become increasingly important.

I. Data Exchange Templates

Corresponding Network Implementation Plan Milestone:

The Board will develop Template guidelines and best practices checklist in 2002. A second version will be released at the end of 2003.

(*"Developing Data Exchange Templates"*)

For the purpose of this document, the terms Template and Schema are used interchangeably. Templates describe and enforce the format and specific restrictions of the data exchanged. They identify what types of information are required for a particular document (i.e., name, address, etc.), as established in predefined standards or agreements. The Beta Phase implemented Templates co-developed with the IMWG Facility Data Action Team (FDAT). These Templates were expressed using the XML Schema Specification. Although complex and new to most participants, the XML Schema Specification was approved as a W3C[32] Recommendation in May 2001. Both SOAP and WSDL, discussed in "Beta Data Transport and Exchange," are implemented using Schema.

While refinement and use of robust Templates was always an important sub-objective of the Project, Template refinement and iterations proved to be intensive and time consuming, but ultimately, highly informative. Consequently, the Beta Phase participants developed numerous recommendations and a Best Practices Checklist for the Board to consider in its development of Template guidelines.

[32] **W3C** – World Wide Web Consortium is an industry consortium that promotes standards for the evolution of the Web and interoperability between WWW products by producing specifications and reference software.

Beta Phase Templates (Schemas)

The Beta Phase used Templates (Schemas) to structure and express all messages processed by Nodes. This included the service requests and responses, as well as the embedded facility data. Beta Phase participants used Schemas to know the structure of messages they should expect and allow the automated validation of request/response against its source Schemas.

Even with only three service request types and relatively simple data, the Beta Phase required several different Schemas:

- Node Service Request Schema: structured the three allowed service requests.
- Component Service Response Schemas: nine Schemas used to structure the data for the *By Change Date* service request (Facility Site, Environmental Interest, Alternative Name, Mailing Address, SIC codes, NAICS[33] codes, Individual, Organization, and Geographic Coordinates). These Schemas were normalized for loading into the EPA FRS table structure.
- Beta Abbreviated Facility Service Response Schema: an abbreviated version of the consolidated Schema used to structure the data in response to the *By Parameter* service request.
- Beta Consolidated Facility Service Response Schemas: a consolidated version of the Beta Component Service Response Schemas that included (by reference) the nine component Schemas. This was used to structure the data for the response to the *By ID* service request. This Schema provides a consolidated "snapshot" of all data on one facility in a single hierarchical file.

Corresponding Network Implementation Plan Milestone:

In 2002, the Board (TRG) will develop guidelines for representing data standards in Templates.

(*"Developing Data Exchange Templates"*)

Table 14, below, details the Beta Phase Schemas.

Table 14. Beta Phase Schemas

Schemas	File Name	Corresponding Service Request(s)
Node Service Request Schema	NodesServiceRequest.xsd	Service requests
Component Service Response Schemas		
– Facility Site	FacilitySiteList_v_1.0.xsd	Service response *By Change Date*
– Environmental Interest	EnvironmentalInterestList_v_1.0.xsd	Service response *By Change Date*

[33] **NIACS Code** – North American Industry Classification System is a system of numerical codes designed to create uniform descriptions of business establishments. This system is in the process of replacing SIC Codes (see definition below).

– Alternative Name	AlternativeNameList_v_1.0.xsd	Service response *By Change Date*
– Mailing Address	MailingAddressList_v_1.0.xsd	Service response *By Change Date*
– SIC codes	SICCodeList_v_1.0.xsd	Service response *By Change Date*
– NAICS codes	NAICSCodeList_v_1.0.xsd	Service response *By Change Date*
– Individual	IndividualList_v_1.0.xsd	Service response *By Change Date*
– Organization	OrganizationList_v_1.0.xsd	Service response *By Change Date*
– Geographic Coordinates	GeographicCoordinatesList_v_1.0.xsd	Service response *By Change Date*
Beta Abbreviated Facility Service Response Schema	OutputSchema_ByParameter_12.14.01.xsd	Service response *By Parameter*
Beta Consolidated Facility Service Response Schema	OutputSchema_ByID_12.18.01.xsd	Service response *By ID*

Table 15 shows one of the Beta Phase Schemas: the Beta Abbreviated Facility Service Response Schema. As indicated above, FDAT derived the tag names for the data element names in the Facility Data Standard used for all Beta Phase Schemas.

Table 15. Beta Abbreviated Facility Service Response Schema

```
<?xml version="1.0" encoding="utf-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:annotation>
    <xsd:documentation>
    This is the abbreviated facility Schema for the Network Node Pilot
    Project - Beta Phase.  This is derived from the facility Schemas
    that were developed for the Facility Data Action Team.  These
    Schemas have been altered for the Beta Phase.  Specifically, many
    data elements, simple type definitions, and element groupings have
    been removed.  Facility identification data includes Site
    Information and Environmental Interest information.
    </xsd:documentation>
  </xsd:annotation>
  <xsd:element name="FacilitySiteList">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="AbbreviatedFacilitySite"
maxOccurs="unbounded"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="AbbreviatedFacilitySite">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="FacilitySiteName" type="xsd:string"/>
        <xsd:element name="StateUSPSCode" type="StateCodeType"/>
        <xsd:element name="LocalityName" type="xsd:string"
minOccurs="0"/>
        <xsd:element name="LocationZIPCode" type="xsd:string"
minOccurs="0"/>
        <xsd:element name="FacilityRegistryIdentifier"
type="xsd:string" minOccurs="0"/>
        <xsd:element name="EnvironmentalInterestType"
        type="xsd:string" maxOccurs="unbounded"/>
      </xsd:sequence>
      <xsd:attribute name="StateFacilityIdentifier" type="xsd:string"
use="required"/>
    </xsd:complexType>
  </xsd:element>
  <xsd:simpleType name="StateCodeType">
    <xsd:restriction base="xsd:string">
      <xsd:length value="2"/>
      <xsd:pattern value="[A-Z]{2}"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:schema>
```

Using the Registry/Repository to Access Schemas

During project development, participants had to receive Schemas via email in order to access them. This presented the predicable problem of version control as the Schemas evolved. When

Schema changes were no longer needed, the participants used the prototype Registry established by the INSG in partnership with National Institute of Standards and Technology (NIST)[34] to store the most current "production" Schemas. Each Schema was given a unique address, and Beta Phase participants simply needed to reference the address in the headers of the XML documents to get access (via HTTP) to the Schemas. The Registry only allows registered users to post and modify the Schemas. Any participant, however, can read and use them.

> **Corresponding Network Implementation Plan Milestone:**
>
> In 2002, the Board will designate a responsible party for overseeing Registry operations, including security, coordinating Registry efforts and publishing Registry guidelines.
>
> (*"Operating and Supporting the Network Registry/Repository"*)

The prototype Registry is scheduled to be replaced by a more permanent Registry later in 2002. Ultimately, the Registry will be used as the official repository for Schemas, including those under development. The benefits of XML, and consequently the Network, depend on a significant number of Partners using the same registered Schemas for data exchanges.

Mapping Templates to Existing Systems and Validating the Results

Schemas are powerful tools to enforce data structure and content. The initial versions of Schemas several elements were restricted to specific enumeration lists that specify allowable values for a given element. While ultimately, these kinds of restrictions will be an important component of ensuring data quality during transfer, they rely on additional conventions about how and when such restrictions should be applied, and should factor-in data-specific information (e.g., States don't all collect the same data or the data that EPA collects), as this kind of information is essential when developing Schemas. Without the benefit of these conventions (whose development is proposed as a Beta Follow-on activity), the Beta Phase initially ran into difficulties as the practical issues of mapping many different source systems to one comprehensive Schema were confronted. Differences in the handling of "missing" or "unavailable" data (neither of which proved to be useful terms) caused validation errors when data for those fields was not available in State agency systems. Of course, handling the lack of such data is a common data interchange challenge.

To test the validation process itself, participants elected to target their constraints to a limited number of key fields. This allowed validation to proceed successfully for most messages, while still flagging those with "fatal flaws." For instance, *By Parameter* requests containing invalid parameters (e.g., searching for environmental interests type "foo") would be rejected.

Participants encountered similar problems with the enumerated-lists fields, especially for environmental interest data.

---

[34] See http://www.metadata.epa.nist.gov.

Lessons Learned and Issues

- Template refinement and iterations proved to be time intensive and time consuming, but also highly informative.
- There are different "right" ways to design and implement Schemas. Network Schema developers must make sure that the Schemas they develop are appropriate for Network Partners.
- Network Schemas must evolve to include conventions for the handling of "missing" or "unavailable" data, and the use of shared enumeration lists for key data. Schemas need to be developed by State and EPA teams working together.
- Several versions of core Schemas will probably have to be developed, each implementing a different level and scope of restriction on its contents. Implementers could then choose (and/or further refine) those Schemas as dictated by their requirements.
- Schema version control is important, but difficult. Proper use of the Registry can ensure coordination of these versions.
- Given that use of common standards across participants is a cornerstone of the Network, guidelines and practices must be developed to jointly manage the "enumeration lists" for key fields, derived from their parent data standards (where those standards exist). Although the participants successfully refined and mapped to a single enumerated list of environmental interest types (despite a diversity of local names for those interests), the handling of "other" types proved difficult. As in the case above in the Exchange Protocol, participants recommended that a Network-wide convention for handling these situations be developed.
- Given its central role, testing of the Registry in its role as a "real t
approved Schema was an important objective of the Beta Phase. The NIST Registry, however, didn't have the full functionality of the eventual Network Registry. For instance, it did not allow participants to alter Schemas once they were registered; if changes were needed, participants had to register the Schemas again after the changes had been made. Despite the limitations of the current Registry prototype, the Beta tests were successful.
- Participants experimented with several alternatives for "missing" or "unavailable" data (including not passing those tags and use of a nillible field), but recognized that a broader solution should be developed for use across all Network Schemas. This report includes a recommendation to this effect, calling for a broad solution to be developed as part of the Network Exchange Protocol to be commissioned by the Board, as outlined in the *Network Implementation Plan*.

J. Beta Phase Experience: Connecting to Existing Information Systems

The Nodes received data from the existing systems that Beta Phase participants use to store and manage environmental data. (While not attempted in the Beta Phase, future requestor Nodes will send data directly into a Partners' information systems.) Partners' information systems vary in complexity and architecture and will have mixed architectures of integrated, federated, warehoused, and isolated systems. These differences are most important when they influence where and how these systems are connected to the Node. Figure 16 depicts some of these architectures and several possible connections to Nodes.

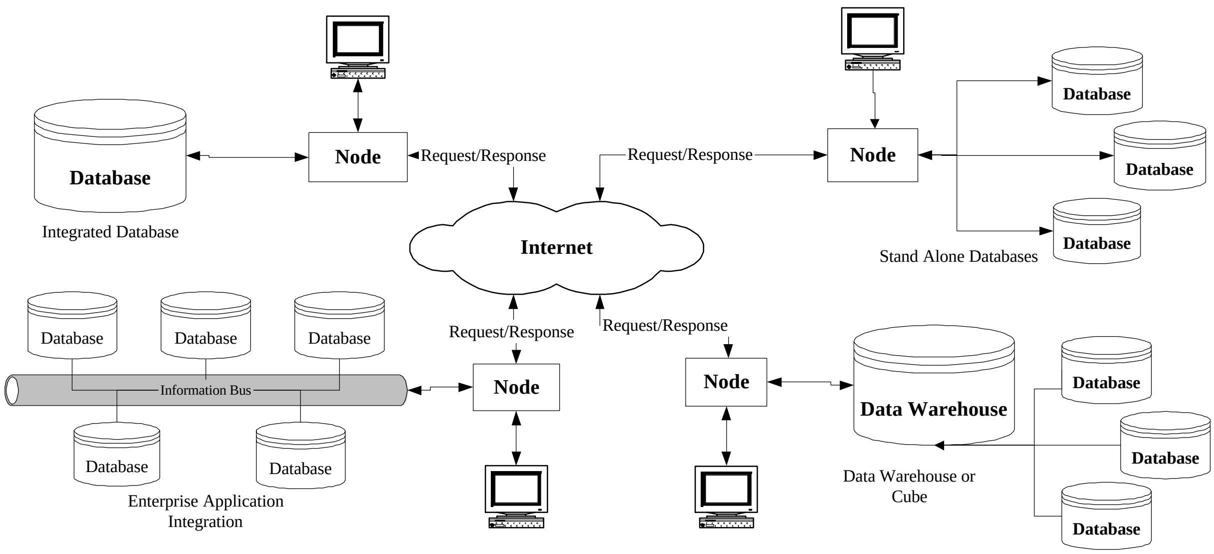


Figure 16. Potential Information System Architectures and Their Possible Relationships to Nodes and the Internet

Each "connection" has two layers: first is establishing the basic "system-to-system" connectivity of the Node to the database (e.g., through ODBC). This connection needs to be made for any applications integration effort. The second layer is the "data mapping." Mapping is the matching of the data defined in the Schema with the data in the existing information system and defining any required translation. Inevitably, differences exist between requested and collected/stored information. Careful definition of data to be exchanged is an integral Network requirement and is enforced by the Templates. (See also, "Data Exchange Templates")

Further, with any enterprise integration effort, connecting a Node to a small number of common systems is easier than connecting a Node to a large number of heterogeneous systems.

The Beta Phase demonstrated that existing application integration tools, such as ODBC, work for Nodes. Regardless of the diversity of hardware and software used by a Partner, Nodes can be established and successfully flow data. In the Beta Phase, participants used three different existing information systems, DB2 (IBM), Oracle 8i/9i (Oracle), and SQL Server 2000 (Microsoft). This experience, because these systems are similar to those used by most other State agencies, suggests that most Partners' existing information system(s) (that contain reliable data, and are connected to a network) can be connected to the Node. Table 16 details the Beta Phase State information systems.

Table 16. Beta Phase State Information Systems

State	Database Description	Platform	Node to DB connectivity	Query Method

DE	Expanded FITS II database-centric, n-tier architecture. SQL Server 2000 Enterprise Edition. In the process of building a single integrated environmental information system. The system is expected to house 100,000 facilities.	MS 2000 advanced server	SQL ODBC	Stored Procedures
FL	40 GB Oracle RDBMS Enterprise Edition 9.0.1.2. Nine Tables with records ranging from 0 – 199,310. Almost FITS compliant. Some tables had to be denormalized to facilitate queries.	Oracle	Oracle 9i Native connection	Uncompiled SQL Queries
NE	DB2/400 running on an AS/400. The Agency has an Integrated Information System(IIS) that supports only one facility record per site and all environmental interests. The IIS contains about 300 normalized physical files (databases). Expanded FITS II model. About 26,000 facilities with over 60,000 environmental interest records.	DB2	JDBC	Uncompiled SQL Queries
NH	4.3 GB Oracle 8.0, Fox Pro, and Access databases. The NH model is FITS compliant and can accommodate FITS II. The Oracle DB is integrated with eight databases and contains 24,700 records.		Oracle 8i ODBC	Stored Procedures
NM	NM is implementing the TEMPO integrated environmental information system based on Oracle 8i. TEMPO is designed to be FITS compliant. Once fully implemented, NMED's TEMPO data will contain approximately 23,000 facility records. For the Beta Phase, the NMED Node had access to a subset of the NMED TEMPO database containing approximately 2,000 test records.	Oracle DB. State servers UNIX and MS Win NT. Node on Linux	Oracle ODBC	Uncompiled SQL Queries
UT	Oracle 8.1.7 database of approximately 13,000 regulated sites. Database is "semi-warehouse" of data extracted from legacy program databases. Uses 51 data elements for high-level description of regulated facilities. Data model closely approximates the FITS II model.		Oracle 8i ODBC	Stored Procedures

Node to Database Connectivity

Part of the technical challenge during Node development was connecting the Node to the existing system. These challenges, however, are not unique to the Network. Connecting a Node to an existing system presents the same challenges as any application integration (e.g., challenges in server software and driver configurations). It is likely that most Partners have used the connectivity tools that they would use to connect their Nodes (e.g., Java Database Connectivity (JDBC)[35], ODBC) in other application integration projects. The added complexity, however, was that the Node-side software (middleware) was unfamiliar to State staff.

Querying the Databases

Once the database is successfully connected to the Node, the Node is able to process the query for the service request. Beta Phase participants took two basic approaches in querying the existing system(s): stored procedures and uncompiled SQL queries. A stored procedure is nothing more than an SQL statement stored inside a database. Stored procedures are compiled

[35] **JDBC** – Java Database Connectivity is a connectivity tool that lets developers using the Java programming language gain access to a wide range of databases and other data sources, either directly or through middleware.

by the database one time: when they are entered. To run the query, the Node simply 'calls' the stored procedure and the database returns the requested data set. This should result in faster database executions and overall performance improvements, but results are less flexible with respect to query changes. Table 17 contains an example of a DNREC stored procedure from the service request type *By Parameter*. Of greatest interest to non-programmers is that this is the place where part of the mapping process occurs from the local database fields to the tag names (data elements) in the Schema. These can be identified by the "select" statements and lines containing

Table 17. Sample Beta Phase Stored Procedure

```
CREATE      Procedure GetFacilityByParameter
     @FacilityName varchar(80),
     @EnvironmentalInterestType varchar(100),
     @StateAbv char(2)='DE'
As
set nocount on
--Check if this request is for Delaware
Declare @FacID int
IF @StateAbv is Null or @StateAbv=''
     Set @StateAbv='DE'
If @StateAbv='DE'
     Set @FacID=0
Else
     Set @FacID=99999999

IF @FacilityName is Null or @FacilityName=''
     Set @FacilityName='%'
If right(@FacilityName,1)!='%'
     Set @FacilityName=@FacilityName+'%'
IF @EnvironmentalInterestType is Null or @EnvironmentalInterestType=''
     Set @EnvironmentalInterestType='%'
--Check for unpermitted query that will return all facilities
If @FacilityName='%' and @EnvironmentalInterestType='%'
Return(-1)

Select         Cast(p.FacID as varchar(11)) as StateFacilityIdentifier,
             FacName as FacilitySiteName,
              isnull(l.StateAbv,'NA') as StateUSPSCode,
             PO_Name as LocalityName,
            l.ZIP5 as LocationZIPCode,
           FacFederalID as FacilityRegistryIdentifier,
           EPAStdName as EnvironmentalInterestType
From tblProgInterest p inner join tblFacility f on
            p.FacID=f.FacID
              inner join tblLocation l on
            f.LocID=l.LocID
              inner join tblZIP5 z on
l.ZIP5=z.ZIP5
              inner join tblPiType pt on
pt.PiTypeID=p.PiTypeID
Where FacName like @FacilityName and EPAStdName like
@EnvironmentalInterestType
And f.FacID>@FacID
Order by FacName
```

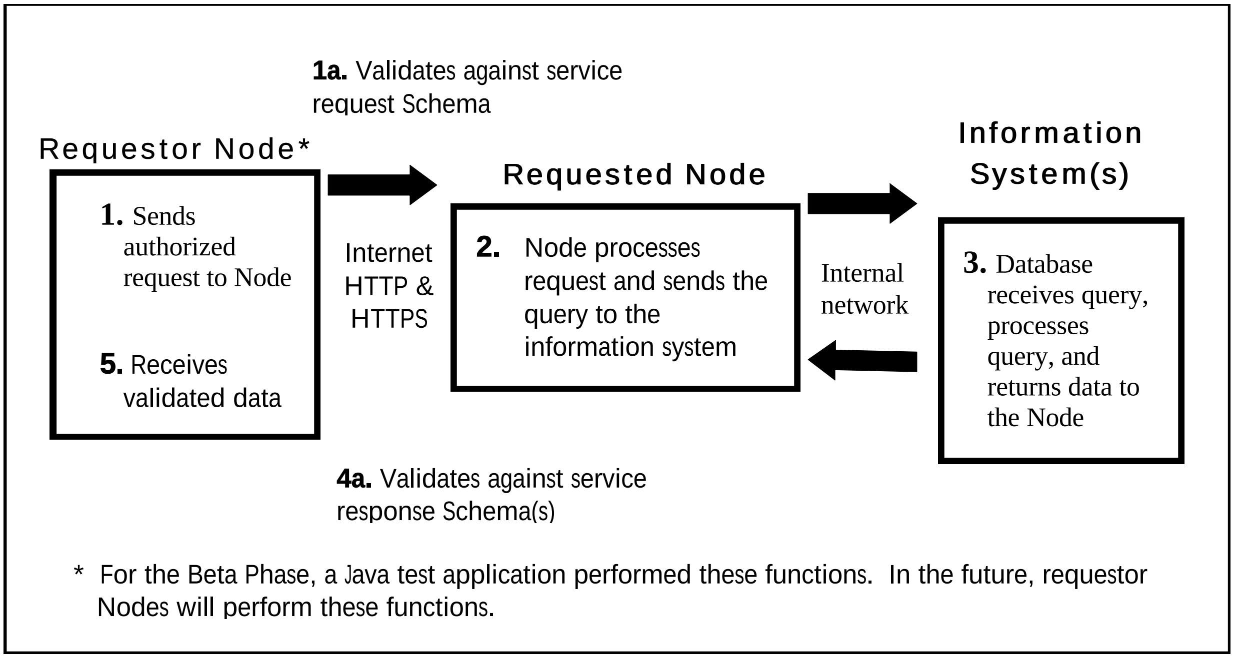
- Existing application integration tools work for connecting Nodes to a wide variety of existing information systems. The diversity in systems represented across States in the Beta Phase is probably greater than any one agency will face internally.
- Because the Node represents a new kind of application, communication between a Partner's Node team and their data administration team is crucial.
- Partners may want to explore further the most effective way of querying data from their database(s). The Beta Phase tested querying with uncompiled SQL statements and Stored procedures. Future tests to determine how to support more flexible information requests are needed.

K. Beta Node Performance

The Beta Phase included two Node performance evaluations:

I. Performance as measured by total transaction time. See Figure 17. The total transaction time has five steps: 1) A requestor Node (or in the case of the Beta Phase, the Java test application) sends a service request to the requested Node. 2) The requested Node processes the service request and sends the resulting query to the information system. 3) The information system processes the query and sends the data back to the requested Node. 4) The requested Node then processes the data and sends the service response back to the requestor Node. 5) The total transaction time ends when the requestor Node receives the service response.

Figure 17. Steps Included in Total Transaction Time



II. <u>Performance as measured by Node processing time.</u> See Figure 16. The Node processing time has three steps and is a subset (steps 2-4) of the total transaction time (described above).

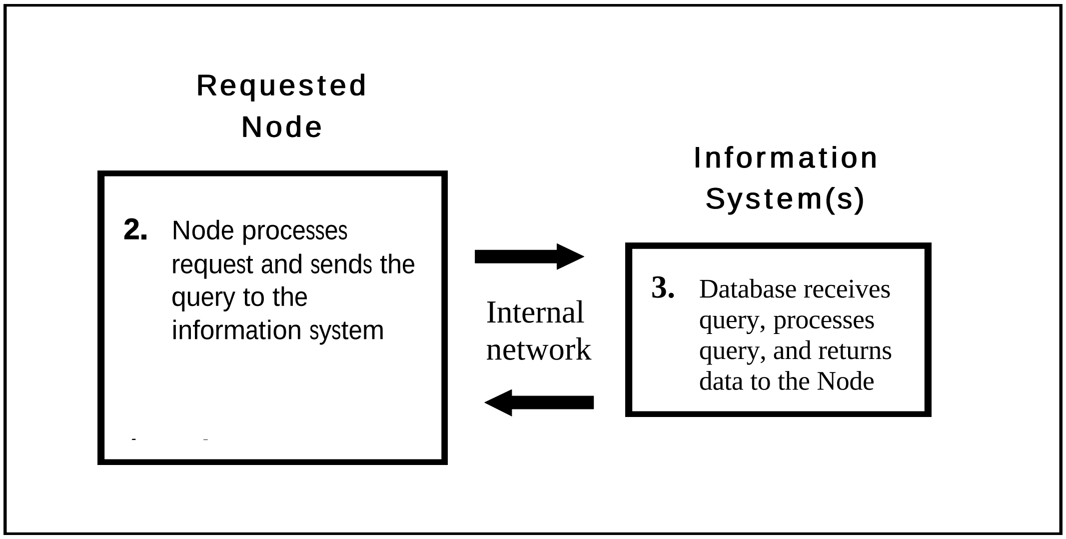


Figure 18. Steps Included in Node Processing Time

Performance Results

Scalability is how well a solution to a problem will work when the size of the problem increases or decreases. Node scalability and speed are inextricably linked. In the case of Node performance, scalability is measured by transaction times at different service request sizes.

To test speed and scalability, each Node processed record sets of different sizes, and the times were recorded. The speed of processing small data sets (less than 3mb) was excellent for all Nodes, each Node able to fulfill this request in seconds or a few minutes. The larger a file becomes, the more the differences in the architecture and set-up of State information systems became a limiting factor. In one case, when processing the service response (since the XML files are built in cache and not saved to the hard drive as they are being built) large files caused the Node server to run out of memory.

Another situation that caused performance degradation (as measured by speed) was when requests for data required many queries. For instance, for one participant, populating information for the *By ID* request for only one facility with four environmental interests, the job had to run 19 separate queries. This complexity and the structure of the existing system play a role in the scalability (transaction database versus report database). These scenarios are not unique to Node transactions: database administrators deal with this whenever databases need to be queried. Optimizing a database for Node queries is no different than optimizing a database for reporting.

Performance Implications for Implementing SSL (Network Security Level 2)

SSL allows for server authentication and data encryption. Beta Phase participants passed all secure requests (SSL enabled) on Port 443, and all unsecured requests on Port 80. Although passing secure information took longer, the difference in time appears very small and manageable.

To test this, five successive and identical queries were run on each port. The times were then averaged and a standard deviation calculated. This test was repeated using a file that was almost one MB, using *By Parameter* service request type, and a file that was 10kb, using the *By ID* service request type. Table 18 provides the test results.

Table 18. Network Security Level 1 (Port 80) versus Network Security Level 2 (Port 443)

	Average Time* Port 80	Standard Deviation Port 80	Average Time* Port 443	Standard Deviation Port 443
Test 1 (1300 records) 800KB *By Parameter* query	17.23 Seconds	± 1.66 Seconds	20.30 Seconds	± 2.61 Seconds
Test 2 (1 record) 10KB *By ID* query	1.11 Seconds	± .21 Seconds	1.57 Seconds	± .46 Seconds

*Note that the times used were total transaction times: from the Java test application to Node and back.[36]

Factors that Influence Performance

The following factors will affect performance:

- The Network/existing information system traffic will influence request/response speed and efficiency.
- Internet connection speed/traffic: Internet congestion will affect the amount of time it takes to service the request. Further, the slower the Internet connection speed, the longer it will take to service the request.
- File Size: the larger the file, the longer it will take to service the request.
- Processor hardware: Beta Phase experience showed that certain hardware issues, such as server memory, could influence performance.
- Database structure and tuning.
- Query method and strategy.

Lessons Learned

- Response time from the existing system is likely to dominate overall Node response time. The same approaches used for optimizing any query/reporting system (i.e., tuning, denormalization, or warehousing) are applicable for improving Node response where

[36] These performance tests only cover relatively small file sizes, however, an implicit design assumption of the Network is that current batch transactions that produce large data sets would be accommodated on the Network by increasing the frequency of transactions, thereby reducing both delays and file sizes. The transaction requirements of some Flows may not be amenable to this approach. In those cases, other performance solutions may include compression or alternative transport protocols.

necessary. To improve performance, run as few queries as possible against production data.

- The Beta Phase participants experienced small, but manageable, performance differences between secure and unsecured data exchanges.
- Many external (non-Node) factors such as Internet, network and existing system congestion, existing system hardware, database structure and tuning, and query method and strategy affect performance.
- Partner's understanding of their own information system and how to best retrieve information can affect performance.

III. Beta Node Planning and Management: Lessons from the Field

L. High-Level Overview

Node establishment, like other information technology projects, requires a coordinated approach and development methodology. The factors behind the project ease or difficulty depend on resource availability, in-house expertise, and external influences outside of the project staff's control. Establishing a Node is no different.

The Beta Phase demonstrated that Partners can build Nodes now. The Beta Phase lessons learned and recommendations, coupled with the (recommended) work to be done in the Beta Follow-on Project, will help make Node establishment easier for future Node builders. However, it is worth noting that every Node establishment will vary, and even if a perfect "how to" Node guide were provided to all Partners prior to Node establishment, there would always be the potential for any number of challenges to arise.

M. High-Level Lessons Learned

The Beta Phase high-level lessons learned regarding Node establishment also apply to most application integration efforts. They are as follows:

- Establishing a Node will be easier once the Functional Specification and Network Exchange Protocol have been developed (and tested and are ready for use). The sooner these are ready, the better for Network Implementation overall.
- The staff establishing the Node (or maintaining/managing after initial establishment) need to have some understanding of Network technologies.
- Support (e.g., resources and authorization/clearance) for the Node from senior decision-makers and managers is important for timely and efficient Node implementation.
- Key staff, who are often located in different areas, will need to coordinate to support Node implementation.
- Outside technical support is likely to be needed for initial Node establishment.
- Given the newness of some of these technologies, outside support may be difficult to find at first.
- There is no such thing as one Demonstrated Node Configuration that "fits all," and every Partner will have more than one suitable configuration. Similarly, no Node configurations used to date have been found unsuitable.
- Implement incrementally.

- The lessons learned from the Beta Phase and the results of the proposed Beta Follow-on Project will help future Node establishers.

N. Approaching Node Establishment: Planning and Preparation

This section describes the lessons learned about Node planning and preparation in more detail.

> Node Beta Phase and work done by the "Follow-On" effort). Following versions will be published between 2002 and 2003.
>
> (*"Establishing Network Nodes"*)

- Establishing a Node will be easier once the Node Functional Specification and Network Exchange Protocol are available (tested and ready for use). Until then, Partners can (and will) establish functional Nodes, but ambiguities about detailed expectations and Node responsibilities will exist. The resulting implementations will be less efficient as Partners spend staff hours and resources on resolving ambiguities. Further, these early implementations will be more susceptible to the interoperability issues that Functional Specification and Exchange Protocol will help to prevent and solve by clarifying expectations.
- The staff establishing the Node (or maintaining/managing after initial establishment) need to have some understanding of Network technologies such as XML, SOAP, web services, XML Schemas, and WSDL. Those planning on establishing a Node would benefit from both general training in these technologies and vendor-specific training, using the tools selected for Node implementation. ("Appendix 2: Issues to be Addressed by Other Parties" recommends that the Board sponsor such trainings.)
- Support for the Node(s) from senior decision-makers and managers is important not only to supply adequate resources to Node establishment, but to overcome potential institutional barriers, such as allowing ports to be opened on agency firewalls. The sooner that support "from above" is communicated, the better.
- As an applications integration project, Node development will involve many staff who are often not in the same location. Database managers, network administrators, and web developers may all need to work together to create a Node that functions properly and can be supported within the Partner's computing environment. The Schema mapping phase of establishing a Flow requires the involvement of staff intimately familiar with both data and the local technologies.
- Partners are likely to need outside technical support for initial Node establishment, especially where a new toolset is being used. This is particularly true because many software companies (both large and small) are currently developing new, relevant tools and upgrading previous versions. On-site technical support will not be needed in all instances: several Beta Phase participants used extensive (but off-site) support for configuration and testing. For example, armed with a complete Functional Specification,

Exchange Protocol and Demonstrated Node Configuration, Partners with adequate in-house technical capacity will not need outside support.

Highlight 1. Take an Incremental Approach (by Dennis Burling)

"Establish your node by proceeding in small incremental steps. With each step, conduct a test to see if that component of the process is working. These small steps make it much easier to troubleshoot any problems along the way… If you try to install everything, then finding the problem is not as simple. This goes for the stored procedures as well. It is much easier to add a few components at a time, then to try them all at once. In our process, the middleware was mapped to an access database first to test the process on the standalone server. Once the process worked on the server, then we started the process to hit against the production data. This was similar to the Alpha process of incremental steps. **Keep it simple: this does not have to be difficult."**

- **Dennis Burling**, Nebraska Department of Environmental Quality

- There is not likely to be one Demonstrated Node Configuration that "fits all." Every Partner will have many tool options. Similarly, no Node configurations used to date have been eliminated as unsuitable options. Tools have different functional options and range in both complexity and price. Many options are likely to have functional capacity far beyond what will be required by the Node Functional Specification. These additional capabilities may or may not be useful to the implementing agency. When considering specific tools, Partners should also consider the following additional factors:
    - The configurations and types of a Partner's existing information systems. For instance, many Partners will need to select a tool that fits within their enterprise information system architecture.
    - The Partner's future information management needs and plans. Partners planning significant infrastructure upgrades should incorporate that strategy into Node tool selection.
    - The Partner's technical capacity. If the Partner does not have in-house expertise in Node technologies and does not want to rely on outside expertise to establish a Node, tools with user-friendly "wizards" and automated functions may be best.
- Finding the "right" technical support may be difficult. Node technologies are new and developing so rapidly that there are relatively few well-qualified technical support contractors to choose from. For similar reasons, what distinguishes one contractor's qualifications from another is not always clear. Recommendations from those who have already established Nodes would be helpful. (Note that the Beta Phase experience is not enough to go on for providing strong recommendations.)

- Implement incrementally. Beta Phase participants found that incremental Node installation was more efficient and successful. (See Highlight 1).

O. Approaching Node Establishment: Node Implementation

Establishing Nodes requires a coordinated approach and development methodology similar to other information technology projects. This report does not provide a systematic "How-to" guide. Detailed literature already exists concerning project management and implementation. It does not make sense to reinvent the existing literature. The following general information is a brief discussion on Node implementation.

> **Corresponding Network Implementation Plan Milestone:**
>
> The "follow-on" will help to establish four additional Nodes in 2002, with a goal of 35 Nodes which exchange data with EPA by the end of 2004.
>
> (*"Bringing the Pieces Together: Continuation of Network Implementation Pilots"*)

As an example, the first steps should be to assure executive sponsor support and to scope the project. Next, implementers should develop a project plan. The project plan should be a dynamic document available to all participants. Next, implementers should develop a communication plan to determine the other people and groups that need to be informed, involved, and participating in decision-making. This will include locating and determining the correct Template. As Node development begins, Partners will need to analyze their current hardware and software situation and assess future needs to determine Node requirements. Concurrently, Partners will need to develop a data mapping design that, in addition to outlining the data mapping strategy, accounts for missing, unavailable, or incomplete information and defines any format translations. Once the Node requirements are established, the Partner must develop a Node design. The Node design should include hardware, software, and existing system needs and configurations. The next step is a coordinated physical Node installation according to the Node design and the data mapping design. The final step is testing and refining the implementation.

P. Estimating Node Costs

This report provides information on Node costs to help Partners roughly estimate the investments they will need to establish a Node. However, for the following four reasons, the enclosed cost information should not be used as a template for Node cost planning:

1. There are many approaches to building a Node. As standards such as SOAP and XML are built into more new and existing applications, these options expand.

2. Partners will have different requirements for their Nodes in terms of scale and performance, and Partners will have different installed software bases. In many cases, Partners will be able to use existing (or planned) software for most or all Node functions.

3. Partners will vary in how much on-staff expertise with the relevant technologies is available, and how much will have to be developed or out-sourced.

Highlight 2. Node Costs: A *General* Breakdown

(The Costs for each Node will differ. The following information is *not* a Node cost template.)

Node Hardware and Software

- Server(s)*
- Firewall ^
- Additional (IT) Network Components
  - E.g., cables
- Application Server ^
- Middleware*
  - Vendor
  - Edition
  - Number and type of licenses
  - Upgrades
  - Service/Maintenance Agreements
  - Etc.
- Database Connectivity
- Security ^

Node Operation

- Initial/First Establishment *
- Consulting*
- Staff Time
  - Gap Analysis/Tool Selection*
  - Maintenance
  - Troubleshooting*

Data Flows (Per Flow)

- Mapping database to Node*
- Query Development (i.e., SQL, Stored Procedures)*

Training

- XML*
- Web Services*
- Software (i.e., Middleware, Enterprise Application Integration)*

* = Costs likely to be much higher for initial Node establishment (and accompanying first Flow)
^ = Most state agencies already have this item or regularly incur this costs related to this item.

4. What costs will be "for the Node" (e.g., purchase of a dedicated box) versus "for a given Flow" or purpose will vary from Partner to Partner. Some Partners may choose to implement their Node by purchasing and configuring a high- performance e-commerce application server and associated hardware to host their Node. Another Partner may elect to use open-source software (e.g., Linux-based tools) running on an existing server and connect it to an existing application. There is no way to usefully compare these two Node costs.

As a cost benchmark, Partners may find it useful to think of their recent experiences with deploying web-based public access (read-only) applications that draw on existing databases. Such projects share nearly all of the software and management issues associated with Node development. Highlight 2 identifies probable cost areas and factors. Some costs, such as data quality, data completeness, and modification of existing systems, may also impact the cost of using a Node to share information.

Estimating Node Costs: The Node Pilot Project Experience

A logical question when gauging Node costs is, "what did the Nodes that have already been established cost?" While this question and its answer are timely, the costs of the Beta Nodes do not accurately represent future Node costs for the following reasons: 1) the Pilot Project was a group "learning" effort on Nodes and for informing future Node establishment, rather than taking existing knowledge and applying it, 2) the Pilot State agencies used only a few tools that are not the most expensive available (e.g., an evaluation version of Microsoft's Biztalk server), 3) the Pilot Project included pilot facility flows rather than complete Network Flows, 4) lessons learned from the Pilot Project are likely to reduce the cost of subsequent Node implementations, and 5) the Pilot had outside support provided by EPA. Keeping in mind these cost caveats, Table 19 shows the Beta Node direct costs.

Indirect costs are not included in the Pilot Node's cost estimates. Indirect costs include tool selection/gap analysis, data quality and completeness, and identifying business requirements. Staff and contractor time will vary depending on technical expertise.

Table 19. Beta Node Costs

Note: These costs exclude contractor costs[1] and do not reflect the expected costs for future nodes (see accompanying text).

State	Hardware	Software	Approx. Staff Hours[2]
Delaware	$3,577	$8,356	40
Florida	n/a	N/A (used their existing enterprise software, plus "free" add-on tools)	80
Nebraska	$3,504	$2,500	30
New Hampshire	$3,500	$1,272 (for a demonstration version of BizTalk)	80
New Mexico	N/A (used existing)	$2,500	80
Utah	Used existing hardware, so no costs were incurred.	Used existing software, so no costs were incurred.	80

[1] For the Beta Phase, EPA provided support for two outside contractors. The costs for hiring these contractors are unlikely to represent future Node contractor costs.
[2] Indirect costs, such as Node tool selection, are not included.

Costs for Each New Flow

Like a firewall or web server, Partners will use Nodes for many Flows and/or applications. Using the Node for a first Network Flow, regardless of the Flow's complexity or size, is likely to require more time (and therefore cost) due simply to the "learning curve." Connectivity to the Node for each additional Flow is likely to cost less than the first, although each Flow will have a unique set of information management needs, such as data quality, data completeness, any additional database or middleware modifications, and assigning responsibility for each of these items. It is important to note, however, that data quality, data completeness, and database modifications are not direct Node costs: they are Flow costs and are present with most information sharing efforts.

Node Costs: Hypothetical Examples

Table 20 contains three hypothetical Node approaches, the assumptions in each approach, and the **estimated ranges** in cost for each approach.   **These are general hypothetical Node costs.** While based on current information, Table 20 (and Highlight 2) should not be used as a proxy for an actual cost assessment.   **Actual Node costs will vary.**

Table 20. General Hypothetical Node Costs

(Initial establishment and extra staff hours for one initial Flow)					
Approach #1		**Approach #2**		**Approach #3**	
Node hardware must be purchased. Non-enterprise low- price solution. Assume you have all necessary accompanying software and hardware. No additional staff training. Consultants only retained for troubleshooting during implementation. Existing information system and data are simple, and staff is capable of mapping to information system.		Node hardware must be purchased. Enterprise mid-price solution. Assume you have all the necessary accompanying software and hardware. Some additional staff training necessary. Consultants retained for implementation only. Existing information system and data are simple, and staff is capable of mapping to information system.		Node hardware must be purchased. High-price enterprise solution. Some additional staff training necessary. Consultants retained for implementation and maintenance and operation. Existing information system and data are complex, and staff is capable of mapping to information system (increase in staff hours).	
Node Costs		**Node Costs**		**Node Costs**	
Hardware	$3,000-$20,000	Hardware	$3,000-$20,000	Hardware	$3,000 -$20,000
Software	$0 -$6,000	Software	$7,000-$17,000	Software	$20,000-$35,000
Consulting	$0 -$5,000	Consulting	$10,000-$20,000	Consulting	$20,000-$30,000
Training	$0	Training	$5,000-$10,000	Training	$15,000-$25,000
Other Direct & Indirect Costs*	$1,000-$25,000	Other Direct & Indirect Costs*	$1,000-$25,000	Other Direct & Indirect Costs*	$1,000-$25,000
Staff Hours**	$8,000-$15,000	Staff Hours*	$5,000-$12,000	Staff Hours*	$7,000-$14,000
Total	**$12,000-$71,500**	**Total**	**$31,000-$104,000**	**Total**	**$66,000-$149,000**

* These costs might include, but are not limited to, backup, disaster recovery, and maintenance.

** Includes extra time for initiating one initial Flow. Note that Flow-specific costs are likely to vary widely.

Q. Beta Phase: Trading Partner Agreements

A Trading Partner Agreement (TPA) defines the information, stewardship, security, and other relevant technical and organizational details essential for mutually agreed upon exchange of information between two or more Partners. In practice, a TPA's length and complexity is determined by the needs of the Partners, the specific Data Flow, and the existence of other governing documents. During the Alpha Phase, the Nebraska Department of Environmental Quality (under the leadership of Dennis Burling, Beta Phase Co-chair) and EPA Region 7 signed the first Network TPA for exchanging facility information with FRS. As of early 2002, one more TPA has been signed, between the Mississippi Department of Environmental Quality and EPA Region 4. These early TPAs have informed the ongoing development of Network TPA guidelines.

> **Corresponding Network Implementation Plan Milestone:**
>
> In 2002, the Beta Phase will complete all documentation, including recommendations for the "Follow-on" effort.
>
> (*"Bringing the Pieces Together: Continuation of Network Implementation Pilots"*)

Initially, the Beta Phase intended to use the expected official Network TPA guidance (that was expected to be complete in the fall of 2001) to develop new TPAs for the Beta Phase. But the TPA development guidance was delayed – the Board is now expected to issue guidelines and checklists in quarter four of 2002. This prompted the Beta Phase participants to put the TPA objective on hold, and ultimately, participants agreed that signing additional TPAs was not as important as other project objectives.

> **Corresponding Network Implementation Plan Milestone**:
>
> The Board will produce TPA guidelines and checklists in 2002 and 2004.
>
> (*"Defining Mutual Expectations for Specific Data Exchanges: Trading Partner Agreements"*)

The Beta Phase participants recognize the strong need for TPAs, especially to "capture" (and exchange) Partner-specific details, such as differences in how data is interpreted or defined by particular Partners. Until Partners employ additional data (and perhaps metadata) standards, these types of specifics will continue to be covered in TPAs. However, adherence to additional data standards will not eliminate a Partner's need for a TPA: TPAs will still be needed to outline the expectations for adherence to particular standards.

As outlined in the *Network Implementation Plan*, the Board will commission TPA recommendations for use by all Partners. These recommendations will provide a common reference point for questions about TPA creation, as well as Templates and checklists to streamline the development process.

> **Corresponding Network Implementation Plan Milestone:**
>
> By the end of 2004, EPA will establish TPAs for all of its National Systems and with 35 Network Partners.
>
> (*"Defining Mutual Expectations for Specific Data Exchanges: Trading Partner Agreements"*)

IV. Next Steps and Recommendations for Follow-on Work

R. High-Level Overview

As detailed below, Beta Phase participants recommend that the Board immediately commission and launch a four-phase follow-on project. A proposal for the follow-on project follows and concludes this Report.

S. High-Level Next Steps and Recommendations

Based on the experience gained and issues uncovered during the Beta Phase, the Beta Phase participants recommend the following tasks be undertaken to support ongoing Network development. These recommendations, while only a subset of the total number of tasks that need further work, represent those that the Beta Phase participants believe need immediate attention in order for Network implementation to begin as planned. ["See Appendix 2: Issues to be Addressed by Other Parties (Recommendations to the Board)"]

T. Building on the Beta: A Proposal to the Network Steering Board for a Beta Follow-on Project

Overview

Beta Phase participants recommend that the Board immediately commission and launch a three-phased Beta Follow-on Project. Beta Phase participants also recommend that an EPA/State team, with the assistance of strong outside technical expertise, execute the Beta Follow-on work. The Project would have the following primary objectives (each driving one project phase):

1. Develop the draft Node Functional Specification (Functional Specification) and the draft Network Exchange Protocol (Exchange Protocol).

2. Test the Functional Specification/Exchange Protocol in controlled Node implementations/Flows using a small test group.

3. Use these tests to revise and document working versions of the Functional Specification and Exchange Protocol for general use.

Beta Phase participants unanimously agree that development of the Functional Specification and Exchange Protocol are the highest priority work areas. These products will provide essential guidance to Node and Flow implementers. Once the Follow-on Project is complete, the Board would decide how to finalize the Functional Specification and Exchange Protocol.

The project will require roughly $300,000 from the Board for specialized technical expertise, technical project management, documentation support, and training. State and EPA participants are expected to provide staff time and any necessary Node hardware or software.

The Beta Follow-on Project should begin as soon as possible, i.e., as soon as funds can be made available (hopefully April 2002). The Project will take approximately nine months from the project start date if outside factors do not cause significant delay.

It is expected that following the conclusion of the Beta Follow-on, but prior to finalization of the Functional Specification and Exchange Protocol by the Board, additional participants will begin using the Specification and Protocol by implementing “early adaptor” Nodes. Funds for these “early adaptor” Node implementations could come from either State operating funds, One Stop Grants, Readiness Grants, or a specialized Node Challenge Grant.

Background

Based on their first-hand experience, the Beta Phase participants identified two specific group action areas that require immediate additional development. The first group of action areas is addressed in this proposal. The second group is discussed in Appendix 2.

Two related factors drive the proposal design:

- The Pilot Project has demonstrated Node functions and their underlying technologies. However, the Pilot also identified many areas where additional conventions regarding the Network-specific implementation of these technologies are needed. Once completed (through this proposed Project), the new conventions will make Node development substantially more efficient and effective.
- Many parties will begin Node planning and implementation within the next several months. They will need these conventions early to prevent duplication and re-work.

This proposal is consistent with the milestones and overall approach in the *Network Implementation Plan* (Several of the milestones depend upon achieving the work outlined in this proposal).

Follow-on Project Objectives

Beta Phase participants have identified the following Project objectives (which may need some adjustment as Project planning develops):

1. Develop, test, document, and publish a draft of the Network Exchange Protocol and the draft Node Functional Specification to ensure compatibility between Flows (as identified in the Type 1 schedule and grant applications), the Functional Specification, and the

Exchange Protocol. Follow-on participants will work with (though not take the lead on) a separate group that is compiling the master list of Flows.

2. Develop metrics/tools to demonstrate/ensure compliance/interoperability with the Functional Specification and Exchange Protocol, and create guidelines for Partners to determine/validate what Nodes are expected to do.

3. Establish stable, regularly-used "production" Nodes that can be relied upon for testing and demonstrating uni- and bi-directional Flows (within this objective is the establishment of two-way Flows through CDX).

4. Develop initial "How to Establish a Node" guidelines and curricula outline. Support (but not lead) whoever the Board charges with the development of a "Node 101" training course (geared for Node implementers).

5. Develop recommendations on at least three Demonstrated Node Configurations (a.k.a. "Node in a Box") to the level of specificity possible. These recommendations will:
   - **a.** Advise early adopter states on Node tool choices.
   - **b.** Advise the Board if and how they should commission the formal development of specific Demonstrated Node Configurations.

6. Work with at least one Network Flow. The group would prefer to table the discussion on "what data" until after this proposal has been approved (the choice of Flow is less important than simply using a Flow).

7. Involve self-selected Beta Phase States, plus one or two additional States that are already knowledgeable about Network technologies and have already expressed interest in the Follow-on work.

Follow-on Project: Three-Phased Approach

The Beta Follow-on Project would be conducted in three phases. Given the rapid growth of Network activity predicted during 2002, the phases may have to be modified as a number of other Board activities stabilize. The Project design described below allows for changes, however, significant delays should be avoided in order to provide the Project results in a timely manner. The Project's nine-month projected timeline is already ambitious, yet Network Partners are already requesting the products from this project. The proposed phases are:

Phase 1 (Months 1-3): Development of Functional Specification and Exchange Protocol.

- Convene a core group of analysts and outside technical experts to develop the draft Functional Specifications and Exchange Protocol. The importance of expert involvement throughout the Project, but especially during this phase, cannot be overstated. To the extent appropriate, participants will adapt Functional Specifications already in use in other exchange networks, such as ebXML, and related standards in the public health and justice areas. This may require training (supported by the Board) of participants.

- Use the identified Flows (proposed as Type 1 or in grant applications) as the universe over which these generic specifications/protocols would apply. This does *not* include any flow specific analysis or Template development.

Phase 2 (Months 4-5): Testing and Refinement of Exchange Protocol/Functional Specification Using Follow-on Nodes.

- Implement Exchange Protocol/Functional Specification in Follow-on Nodes (probably one Node for each participant).
- Select a small data domain (Flow) for bi-directional Flow to use for testing purposes.

Phase 3 (Months 6-9): Analysis of Test Results and Publication of Draft Exchange Protocol/Functional Specification

- Re-convene core group of analysts, outside technical experts, and participants to assess test results, and revise the draft Functional Specification/Exchange Protocol as needed.
- Conduct second round of additional testing based on draft Functional Specification/Exchange Protocol adjustments.
- Use the test experience to recommend to the Board how development of Demonstrated Node Configurations should be approached.

The Project would conclude with delivery of the draft Node Function Specification, the draft Network Exchange Protocol, recommendations on Demonstrated Node Configurations, and a draft "How-to" Node guide for the Board for final review and dissemination.

Possible subsequent efforts (Month 10-forward) could include:

- Implementation of the revised Functional Specification/Exchange Protocol by a larger group of early adopter Partners who would agree to document their experience. (Presumably these early adaptors would be implementing a variety of Flows over their Nodes.)
- Development and testing (in the early adopter group) of Demonstrated Node Configurations that use the Functional Specification and Exchange Protocol.

Project Participation, Staffing and Resources

As always, the key resource for Network projects will be the participation of its members. CDX and most Beta Phase participants (pending approval of their management) are prepared to participate. Participation of one or two additional State agencies with immediate expertise and interest would be discussed [37].

The next most *critical* resource for the project will be the counsel and technical leadership of one or more technology experts who have as much direct implementation experience with as many of these technologies as possible. The project will also require a technical project manager who will

[37] The Beta Phase participants have already identified potential candidates. Potential candidates include, Michigan, Mississippi, Oregon, Pennsylvania, and Wisconsin.

manage the project as a formal specification and application development exercise. The Beta Phase project was successful, even though nearly all participants (consultants and vendors included) were learning these technologies as they went. The Follow-on Project will need a technical lead that can speak from experience (other than the Beta Phase, though carry-over from the Beta Phase is also essential). All phases of the project will likely require at least 1 to 1½ FTE of staffing support. With the approval of this initial proposal, identifying this expertise should begin *immediately*.

Phases 1-3 of the project would be supported by in-kind contributions from States and EPA, as well as funds from the Board in the amount of $300,000. These funds would support:

- Technical Project Lead staffing
- Technical consultations with XML/e-commerce experts
- Project Manager staffing
- Documentation services
- Software/Hardware configuration support
- Limited direct support to Participants for other expenses incurred on behalf of the project.

Given the complex mix of possible funding sources for this work (EPA and State agency operating funds, Board funds, and possibly some readiness grants and challenge grants), the Board and this Project team will have to balance many factors in determining a final appropriation. However, given the pressing need for the Functional Specification and Exchange Protocol, and the "critical mass" of willing participants already assembled, the Board should place a priority on funding this work in a way that it can begin immediately and proceed efficiently. This may rule out the use of some (non-Board) grant funds for early work.

Next Steps

Participants request that the Board consider this proposal on its call on March 22nd. If the Board approves this initial proposal, the Beta Phase participants (after a short break!) are prepared to oversee Project start-up activities, such as soliciting participants and leadership, and developing a more complete workplan, schedule, and budget for the Board's approval at the next earliest opportunity.